

1 Un nuevo año.... un nuevo reto

Revista de Software Libre ATIX 2009

Colaboradores

Dirección y Coordinación General

Esteban Saavedra López (jesaavedra@opentelematics.org)

Diseño y Maquetación

Jenny Saavedra López (jennysaavedra@gmail.com)

Esteban Saavedra López (jesaavedra@opentelematics.org)

Revisiones

Esteban Saavedra López

Jenny Saavedra López

Noticias

Ivonne Menacho

Jenny Saavedra López

Marcia Velasquez

Autores Frecuentes

Arnold Guzmán

Ernesto Rico Smith

Esteban Saavedra López

Lucy Ayarde Romero

Williams Chorolque Choque

Herramientas

La edición de esta revista fue realizada de forma integra haciendo uso de Software Libre

ATIX

Palabra quechua,
con un sentimiento profundo
y con gran significado filosófico

El que lo sabe

El que lo intenta

El que lo puede

El que lo logra

El que lo sabe, el que lo intenta, el que lo puede, el que lo logra

Editorial

Hay muchísimas creencias cuando empieza un nuevo año, en general se dice que todo lo que se hace el primer día de año, se hará todo el año. Consideramos que lo más importante al empezar un nuevo año es tomarse un tiempo para meditar y evaluar las cosas buenas y malas que tuvimos, aprendiendo de las malas para que éstas no vuelvan a pasar y de las buenas para que éstas se vuelvan a dar, pero de mejor forma. Otro aspecto importante es realizar una planificación acerca de las metas a las que deseamos llegar y los objetivos que deseamos cumplir, todo esto representa un gran reto, que debemos saber afrontar.

Un nuevo año .... un nuevo reto, un título que sugiere de por sí, lo que todo ser humano debe comprender bien, que cada nuevo año enmarca un nuevo reto, donde el mismo, esta formado de retos diarios que debemos afrontar; retos en lo personal y en lo profesional, el cumplimiento de éstos representará cuán bien estemos, y cuanto progresemos.
La revista Atix y sus miembros tenemos un gran reto, será el de consolidar nuestro trabajo y aporte que empezó hace unos meses atrás y que a lo largo de este tiempo nos ha traído grandes satisfacciones, pero estamos seguros que no debemos dejarnos, más al contrario debemos ponerle más esfuerzo y sacrifico, por que también estamos seguros que esto se traducirá en nuevas grandes satisfacciones tanto para los miembros de la revista, como para los autores y nuestros lectores.

En éste sexto número ponderamos un par de aspectos, como:

- ✔ el nacimiento de una nueva sección de noticias breves concernientes al mundo de la tecnología y la información, esta sección lleva por nombre "**Willay news**", willay es una palabra quechua que significa: comunicar, transmitir, notificar, avisar; significado que caracteriza a esta sección por tener como objetivo el transmitir y/o comunicar noticias breves de actualidad. Esta sección esta a cargo de tres miembros colaboradores de nuestra revista: Ivonne, Jenny y Marcia.
- ✔ Empezamos a registrar nuestra historia, mediante un separador que muestra nuestros números anteriores y sus fechas de publicación.
- ✔ Continuamos con varios de los artículos que son del interés de nuestros lectores, agradecemos a sus autores por la dedicación emprendida en los mismos.

Un nuevo año.... un nuevo reto, pon tu mejor empeño para lograr tu reto de este año.

Bienvenidos a nuestro séptimo número

Esteban Saavedra López
Director y Coordinador General

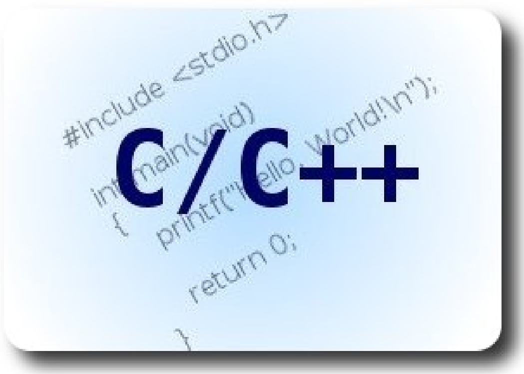
Contenido

Liberado el 26 de enero de 2009



Utilizando herramientas de desarrollo C++ en Linux (3ra parte)

Administrar un proyecto de desarrollo puede llegar a ser un trabajo monótono y extremadamente mecánico si es llevado a cabo manualmente. Los programadores deberíamos preocuparnos en hacer software de calidad, y no gastar el tiempo y esfuerzo ideando nuevas formas de compilarlo.



Bienvenidos a ésta tercera y última parte de las herramientas de desarrollo en Linux. En esta ocasión vamos a hacer una breve introducción práctica a **autotools**, que en realidad son varias herramientas integradas para definir las reglas de construcción de nuestros programas, no importa si éstos son grandes proyectos o pequeñas pruebas.

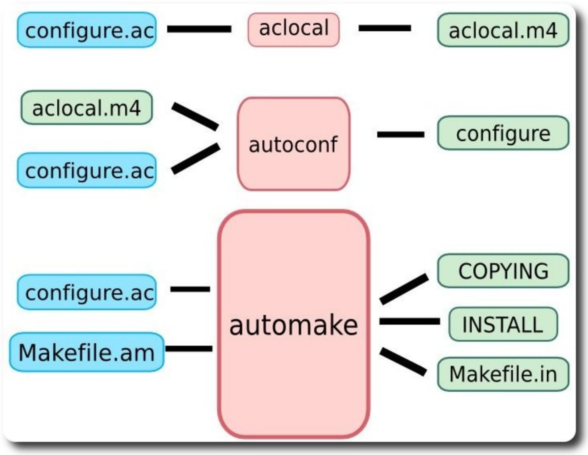
Es importante saber que **autotools**, son herramientas que se activan principalmente desde la línea de comandos, no son un entorno de desarrollo integrado (IDE) como se parecería de inicio.

El conjunto de herramientas **autotools**, está conformado por los siguientes programas:

- ✔ **Aclocal**: Recopila información de que librerías externas y macros soporta **autotools**
- ✔ **Autoconf**: Permite verificar la existencia de características dentro de un sistema en particular, tales como alguna librería, función o algún comando, que son necesarios para que un programa pueda ser compilado correctamente.
- ✔ **Automake**: Por medio de plantillas y apoyado principalmente por el programa **`Make`**, define la forma en que un programa será compilado, aquí es donde se especifican todos los archivos fuente necesarios para el proceso de construcción de programas ejecutables y librerías que se desean generar. **`automake`** también define todos los archivos de configuración, gráficos y otros que necesitará el programa.
- ✔ **Autoheader**: Transfiere la información recopilada por **`autoconf`** para ponerlo a disposición del código fuente dentro de nuestros mismos programas.
- ✔ **Libtool**: Permite administrar la construcción de librerías estáticas y dinámicas de una forma fácil y transparente, sin tener que lidiar con los problemas de una compilación manual.

Un poco de teoría

Cada herramienta tiene ciertos archivos de entrada con los que trabaja, y ciertos archivos que genera. Estos últimos son generados a su vez por otras herramientas en una distinta etapa de construcción. Todas estas están reflejadas en el siguiente gráfico:



Esquema de funcionamiento de **autotools**

Analicemos cada una de estas etapas.

aclocal

La dependencia que existe respecto a cualquier librería externa, requiere información que se pueda conseguir de la misma. Cuando las librerías tienen soporte para **autotools**, éstas poseen ciertas definiciones que vienen en la forma de macros. Por ejemplo, para la librería de desarrollo GNOME en XML: libxml2, podemos encontrar la ruta donde están los archivos de cabecera para utilizar la librería de la siguiente forma manual:

```
$ pkg-config libxml-2.0 –cflags
```

Con esto obtendremos

```
-I/usr/include/libxml2
```

¿Pero para que sirve esto? Bien, recordando un poco el manejo del compilador g++, podemos añadir esta última línea obtenida a las opciones del compilador para incluir la ruta de búsqueda de los archivos de cabecera. Es decir, si queremos compilar algo incorporando **`libxml`** en nuestro programa, debiéramos hacerlo de la siguiente manera:

```
$ g++    fuente.cpp
-I/usr/include/libxml2  -o  programa
```

Este tipo de información se puede obtener de forma automática.

En el anterior gráfico se observa que un archivo llamado **`configure.ac`** es necesario antes de ejecutar **`aclocal`**. Dentro de este archivo, se hace una verificación de la existencia de **`libxml`** (y de cualquier otra librería):

`configure.ac:`

```
PKG_CHECK_MODULES(LIBXML2, [libxml-2.0 >=
2.6.19])
AC_SUBST(LIBXML2_CFLAGS)
AC_SUBST(LIBXML2_LIBS)
```

Básicamente el macro **`PKG_CHECK_MODULES`** se encarga de verificar la existencia de **`libxml2`**, de localizarlo y de compilar la versión mínima requerida (en este caso 2.6.19). Los macros **`AC_SUBST`**, obtienen información de dicha librería (**`cflags`** y **`libs`** son opciones que se pasa al compilador y al **`linker`** respectivamente).

Sin embargo, todos estos macros son incomprensibles, puesto que se deben importar de alguna fuente. El programa **`aclocal`** se encargará de este trabajo. El resultado es el archivo aclocal.m4 donde están las definiciones.

autoconf

`Autoconf` también trabaja con **`configure.ac`**, y, a diferencia de **`aclocal`**, se encarga de generar un script ejecutable llamado configure que hace el proceso de la verificación en el sistema de todas las dependencias. Otro archivo de entrada necesario es aclocal.m4 generado por **`aclocal`**.

Dentro de **`configure.ac`** también se establece el lenguaje de programación a utilizar, la versión del programa, y verificación de funciones particulares, las cuales dan un mayor soporte a la portabilidad del proyecto en el cual se está trabajando.

automake

`automake` permite crear plantillas de lo que será nuestros archivos finales **`Makefile`**, que

a su vez se ejecutarán con **make** y se creará el proyecto. **automake** necesita de los archivos **configure.ac** y **Makefile.am** para hacer su trabajo. Es posible que varios archivos **Makefile.am** existan en cada proyecto, y que cada directorio tenga su propio archivo, para así definir la estructura del proyecto y la estructura de instalación/desinstalación.

automake presenta el concepto de los llamados primarios (**PRIMARIES**), que son sufijos descriptivos acerca de un propósito en específico. Ilustremos algunos primarios de la siguiente forma:

Para definir el ejecutable llamado programa debemos establecer lo siguiente en:

Makefile.am

```
bin_PROGRAMS=programa
El primario en este caso es PROGRAMS, que
establece los ejecutables. A partir de
esto podemos hacer otras definiciones
para programa:
programa_SOURCES=main.cpp
```

Por medio del primario **SOURCES**, establecemos que el ejecutable programa necesita del archivo **main.cpp** para poder construirse. Para integrar nuestro proyecto con alguna librería, necesitamos otros primarios de apoyo:

```
programa_CPPFLAGS=$(LIBXML_CFLAGS)
```

Recuerden que **LIBXML_CFLAGS** fue obtenido por **aclocal** y **autoconf**, y por medio del primario **CPPFLAGS** y será añadido a la lista de argumentos que se pasarán en tiempo de compilación de programa. Todas estas definiciones serán transformadas a lo largo de todo el proceso gracias a **automake,** similar a:

```
g++ main.cpp -o programa  -I/usr/include/
libxml2
```

Recuerden que éste es un ejemplo bastante simplista, y tal vez parezca más fácil teclear la última línea en vez de ejecutar todo el proceso de **autotools**, sin embargo, cuando los archivos fuente se van haciendo muchos, en diferentes directorios definiendo varios módulos, y se utiliza una docena de librerías, escribir la generación a mano de un solo archivo ejecutable puede llegar a ser de muy escaso valor práctico, y terriblemente propenso a errores.

libtool

Crear librerías exige el uso de un conjunto de herramientas propias de cada sistema, ya sean estáticas o dinámicas. **Libtool** permite definir librerías de una forma más simple. Por ejemplo, para definir la libreria **libmylib.la**, se debe hacerlo en **Makefile.am**, por medio del primario **LTLIBRARIES**

```
lib_LTLIBRARIES=libmylib.la
libmylib_la_SOURCES=mylib.cpp
```

Esta librería puede servir para crear pasos intermedios en el proceso de construcción de una aplicación, de esta forma los ejecutables, e incluso otras librerías, podrán hacer uso de las mismas.

Hagamos que **programa** utilice esta librería por medio del primario **LIBADD**:

```
bin_PROGRAMS=programa
programa_SOURCES=main.cpp
programa_LIBADD=libmylib.la
```

Sin embargo, para poder gozar de los beneficios de **libtool**, debemos utilizar la herramienta **libtoolize** antes de llamar a **automake**.

autoheader

Por medio de **autoheader**, es posible poner la información recopilada por las otras herramientas a disposición de nuestros programas. Por ejemplo, dentro de **configure.ac**, uno de los macros de uso obligatorio que activaremos será

```
AM_INIT_AUTOMAKE(programa, 0.2.4)
```

Los dos parámetros del macro **AM_INIT_AUTOMAKE** son el nombre del programa y la versión (en éste caso 0.2.4). Para usar **autoheader** también se debe utilizar el macro:

```
AC_CONFIG_HEADERS(config.h)
```

que establece el archivo **config.h** (que será generado), como aquel donde la información será recopilada. La información que nos interesa está en forma de macros para C, por ejemplo, los dos parámetros anteriores se guardan dentro de:

config.h

```
#define VERSION "0.2.4"
#define PACKAGE "programa"
```

Es posible utilizar dicho archivo como cualquier fichero de cabecera normal de C:

```
#include “config.h”
```

y utilizar dicha información como mejor veamos conveniente.

Un ejemplo práctico: Creando un proyecto

Al igual que en la anterior entrega, vamos a crear un proyecto bastante simple, que consta de dos librerías, cada una con su propio archivo fuente y su cabecera (.h) , y un tercer archivo fuente (**programa.cpp**) que hace uso de éstas dos librerías propias, y la librería **libxml2**. Todos los archivos aquí mencionados irán en un solo directorio. El código fuente se lista a continuación:

libreria1.h

```
int suma(int a, int b);
```

libreria1.cpp

```
#include "libreria1.h"

int suma(int a, int b){
        return (a+b);
}
```

libreria2.h

```
int resta(int a, int b);
```

libreria2.cpp

```
#include "libreria2.h"

int resta(int a, int b){
        return (a-b);
}
```

programa.cpp

```
#include "libreria1.h"
#include "libreria2.h"

#include <iostream>

#include <libxml/tree.h>
#include <libxml/parser.h>

#include "config.h"

using namespace std;


int main(void){
        xmlDocPtr doc; // solo para probar la libreria
        cout << "Bienvenidos a " << PACKAGE <<
                " Version: "<< VERSION << endl;

        int a = suma(1,2);
        int b = resta(4,3);
        cout << " La suma es: " << a <<
                " y la resta es: " << b << endl;

        return 0;
}
```

Primero debemos definir las dependencias del proyecto. La única dependencia es la librería libxml2, definamos el archivo **configure.ac**. Tomar en cuenta que **autoconf** genera el script configure, el cual hace la verificación de todo lo necesario. A continuación su listado:

configure.ac

```
#verifica que el codigo fuente exista
AC_INIT(programa.cpp)
# nombre y version del programa
AM_INIT_AUTOMAKE(programa, 0.2.4)

AC_CONFIG_HEADERS(config.h)

```

```
# indica que el lenguaje es C
AC_PROG_CC

# adicionalmente indica que existe C++
AC_PROG_CXX

AC_PROG_INSTALL # genera make install

PKG_CHECK_MODULES(LIBXML2, [libxml-2.0 >= 2.6.19])
AC_SUBST(LIBXML2_CFLAGS)
AC_SUBST(LIBXML2_LIBS)

AC_CONFIG_FILES(Makefile)

AC_OUTPUT
```

AC_INIT toma como argumento cualquier archivo del código fuente para verificar la existencia del proyecto, de este modo **autoconf** se inicializa. Con **AM_INIT_AUTOMAKE** se inicia el sistema **automake**. Note el sufijo **AM** para las herramientas de **automake**, y **AC** para **autoconf**. **AC_CONFIG_HEADERS** inicializa el sistema **autoheader** como fue explicado anteriormente.

A continuación se debe definir el lenguaje de programación utilizado en el proyecto. Al estar utilizando **C** Y **C++**, debemos especificarlo con dos macros, respectivamente: **AC_PROG_CC** y **AC_PROG_CXX**. **Autoconf** también verificará la existencia de los compiladores para dichos lenguajes en el sistema.

AC_PROG_INSTALL generará el código necesario para hacer una instalación limpia de la aplicación dentro del sistema siguiendo las prácticas normalmente utilizadas de rutas de ficheros.

PKG_CHECK_MODULES verifica la existencia de la librería **libxml2** y su versión mínima. Luego de esto **AC_SUBST** pone a disposición variables que serán de utilidad luego para el compilador y el **linker**.

AC_CONFIG_FILES establece todos los **Makefiles** que se generarán para este proyecto. En nuestro caso solo se creará uno dentro del directorio actual.

AC_OUTPUT generará todos los archivos intermedio necesarios que el conjunto de herramientas **autotools** requiere.

Luego de definir las dependencias, toca definir la estructura del proyecto, como todo está en un solo directorio, es muy fácil de hacerlo de la siguiente manera, creando **Makefile.am**

```
bin_PROGRAMS=programa
programa_SOURCES=programa.cpp libreria1.cpp libreria2.cpp
programa_CPPFLAGS=$(LIBXML2_CFLAGS)
programa_LDADD=$(LIBXML2_LIBS)
include_HEADERS=libreria1.h libreria2.h
```

Primero definimos el nombre del programa ejecutable, con el primario **PROGRAMS**. Nuestro programa se llamara “programa”, de esta forma, definimos la lista de todos los archivos fuente para crear programa con el primario **SOURCES**.

Sin embargo, para generar ejecutables, se necesitará pasarle ciertas opciones al compilador. Dichas opciones las suministra la misma librería (que normalmente comprende la localización de los archivos de cabecera), a través de **LIBXML2_CFLAGS**, que fue registrado previamente dentro de **configure.ac** con el macro **AC_SUBST**. Para pasar las opciones al compilador se necesita el primario **CPPFLAGS**. De modo similar, **LIBXML2_LIBS** son las opciones que se le pasa al **linker** (opciones de localización física de las librerías), definido por medio del primario **LDADD**.

Finalmente los archivos de cabecera locales se definen con el primario **HEADERS**.

Una vez definidos todos los archivos, vamos a empezar el proceso de construcción. Aclarar que se generarán bastantes archivos intermedios, aquí solo mencionaremos los más importantes.

Primero ejecutamos **aclocal** para cargar macros

```
$ aclocal
```

Esto debería generar el archivo **aclocal.m4**. Luego ejecutar **autoconf**

```
$ autoconf
```

Principalmente genera el script configure, con su permiso de ejecución activado. Es el momento de ejecutar autoheader

```
$ autoheader
```

Se debe generar el archivo **config.h.in**, que es una plantilla que configure se encargará de convertirlo en **config.h** después de hacer la verificación de características presentes en el sistema. Luego ejecutar **automake**:

```
$ automake
```

Sin embargo, **automake** no se ejecutará correctamente, ya que saldrá varios mensajes "**required file**", es decir que faltan varios archivos para continuar. Podemos hacer que **automake** genere dichos archivos por si mismo, utilizando la opción –**add-missing**

```
$ automake --add-missing
```

Sin embargo, aun ciertos archivos son necesarios, que deben ser creados manualmente, bajo el siguiente criterio.

NEWS: Lista de releases y noticias acerca del programa.

README: el "leeme", usualmente la primera fuente de información para un usuario nuevo.

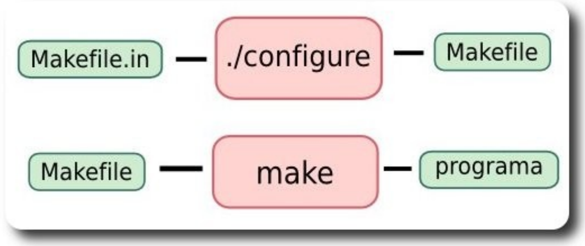
AUTHORS: Nombres de los creadores del programa

ChangeLog: Bitácora de cambios del programa según las versiones

Incluir esos archivos es una práctica estandarizada y casi obligatoria al desarrollar aplicaciones en entornos libres. Si uno no va a hacerlo se debe incluir la opción –**foreign**

```
$ automake --add-missing --foreign
```

Así no saldrá ningún error más. Verificar que se haya creado el archivo Makefile.in, que es la plantilla de Makefile, la cual será procesada primero por configure.



Esquema de scripts generados por **autotools**

Hasta aquí, concluye el trabajo de **autotools**, queda ejecutar el script generado configure, el cual hará una verificación en el sistema de que todas las librerías y demás características estén debidamente instaladas. No olvidar los caracteres "./" al inicio:

```
$ ./configure
```

Dos líneas de la salida son importantes:

```
config.status: creating Makefile
config.status: creating config.h
```

Lo que comprueba que se hayan creado **Makefile** y **config.h**. Teniendo estos archivos, ya podemos hacer que **make** cumpla su trabajo y compile todo nuestro código fuente sin problemas:

```
$ make
```

Si existiese algún problema, es porque el código fuente tiene errores de sintaxis, y el proceso se detendrá ahí. Si todo esta bien, entonces se habrá creado el ejecutable programa. Vamos a probarlo.

```
$ ./programa
```

```
Bienvenidos a programa Version: 0.2.4
La suma es: 3 y la resta es: 1
```

Sin embargo, este programa todavía no está instalado en el sistema. Para hacerlo, **Makefile** nos da más opciones. Utilizar **make install** (normalmente necesita permisos de superusuario):

```
$ make install                    (o sudo
make install)
```

Esto copia el ejecutable en la ruta estándar **/usr/local/bin/**, también copia los archivos de cabecera en **/usr/local/include/** así podemos acceder globalmente a estas características. Análogamente **make uninstall** desinstalará la aplicación limpiamente.

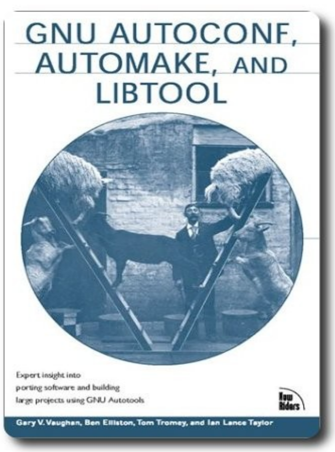
Finalmente, nuestro programa debe estar preparado para la distribución, para esto vamos a empaquetarlo con:

```
$ make dist
```

Esto creará el archivo comprimido **programa-0.2.4.tar.gz**, el cual contiene todos los archivos necesarios para su uso.

Conclusiones.

Si usted desea más información puede consultar el libro "**GNU AUTOCONF, AUTOMAKE AND LIBTOOL**" denominado también autobook o "**the goat book**" (**http://sources.redhat.com/autobook/**).



Portada de autobook

Autotools nos da muchas funcionalidades bastante útiles para hacer programas, sin embargo, puede que tenga una curva de aprendizaje muy pronunciada, por esto que se han creado varias alternativas como **Cmake** (**http://www.cmake.org**), apache **ant**, basado en java y scripts xml (**http://ant.apache.org**) . Usted también puede optar por usar simplemente make.

Referencias

[1] http://sources.redhat.com/autobook

[2] http://www.gnu.org/prep/standards/

Autor



Arnold Marcelo Guzmán

Desarrollador
spacerockganimedes@gmail.com

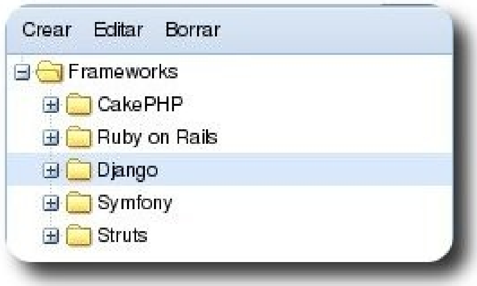
Introducción a Ext JS (3ra parte)

Ext JS es un Framework que permite crear aplicaciones Web 2.0 con interfaces muy similares a la de una aplicación de escritorio.

En la parte 2 de éste tutorial, hemos creado un árbol que nos permite presentar nodos así como moverlos a algún lugar del árbol, ahora les mostrare como editar el texto de los nodos, creación y borrado de nuevos nodos, para esto crearemos un menú que nos permita realizar estas acciones en el nodo seleccionado. Primero debes aumentar un nuevo link en la cabecera que lucirá de esta forma:

```
<script type="text/javascript"
src="ext-2.2/adapter/ext/ext-
base.js"></script>
<script type="text/javascript"
src="ext-2.2/ext-all-debug.js"></script>
<script type="text/javascript"
src="ext-2.2/source/locale/ext-lang-
es.js"></script>
…
Ext.onReady(function() {
    // Estas dos líneas sirven para poder
presentar los mensajes de error
    Ext.QuickTips.init();
    Ext.form.Field.prototype.msgTarget =
'side';
...
```

Para evitar confusiones y puedan seguir mejor las modificaciones del código podrán descargarlo de la siguiente dirección: **http://github.com/boriscy/atixtutorextjs/tree/master**. Lo primero que hemos realizado es aumentar un vínculo javascript para poder seleccionar el lenguaje, en este caso español; luego hemos aumentado dos líneas que nos permitirán presentar los mensajes de error al costado del campo (input). Ahora adicionemos el código que nos permitirá presentar el menú para poder crear, editar y borrar nodos.




```
var tree = new Ext.tree.TreePanel({
 id: 'arbol',
 ...
 tbar :
  [{ text: "Crear", handler: crearNodo
  },{ text: "Editar", handler: editarNodo
  },{text: "Borrar", handler: borrarNodo
  }]
});
```

Aquí aumentamos tres botones de menú para poder crear, editar y borrar nodos, ahora además debemos crear tres funciones distintas que permitirán realizar las acciones de cada uno de estos botones, para poder realizar la edición y creación de nodos. Ahora creamos las funciones que permitan estas acciones, éstas deben estar antes de la creación del árbol e decir antes de la parte del código **var tree = new Ext.tree.TreePanel**, para crear un nodo se inserta el siguiente código:

```
function crearNodo() {
        // Se busca el nodo seleccionado
        var nodo = Ext.getCmp('arbol').getSelectionModel().selNode;
        var w = new Ext.Window({
            title: 'Crear Nodo', width: 250, modal: true,
            items: [{
                xtype: 'form', id: 'formaCrearNodo', labelWidth: 40, url: 'arbol.php',
                items:[{
                    xtype: 'hidden', name: 'accion', id: 'accion', value: 'crear'
                },{
                    xtype: 'hidden', name: 'padre', id: 'padre', value: nodo.id
                },{
                    xtype: 'textfield', fieldLabel: 'Texto', name: 'nombre', id:
'texto', allowBlank: false
                }]
            }],
            buttons: [{
                text: 'Guardar', handler: function() {
                    var forma = Ext.getCmp('formaCrearNodo');
                    // Recuperamos el nombre del nodo
                    var nombre = forma.getForm().getValues().nombre;
                    forma.getForm().submit({
                        success: function(a, resp) {
                            //resp.result.id
                            // Busca el parent Contenedor y lo cierra
                            nodo.appendChild(new Ext.tree.TreeNode({
                                id: resp.result.id,
                                text: nombre
                            }));
                            forma. findParentByType('window').hide();
                        },
                        failure: function(a, b) {
                            // Se muestra un mensaje de error
                            Ext.MessageBox.show({
                               title: 'Error',
                               msg: 'Existion un error al crear el nodo!',
                               buttons: Ext.MessageBox.OK,
                               icon: Ext.MessageBox.ERROR
                           });
                        }
                    });
                }
            }]
        });
        w.show();
    }
```

Lo primero es ver cual es el nodo seleccionado, luego creamos una ventana que presentará una forma, para que solo esté activa esta ventana seleccionamos **`modal: true`**, la ventana solo tiene un ítem que es la forma y esta a su vez tiene 3 items y un botón, el primer ítem es un campo oculto en el cual almacenamos la acción que se realizará, en este caso crear, el segundo campo oculto almacena el padre y el tercero es un campo de texto en el cual se escribe el texto, luego existe un botón que es el encargado de realizar el POST, osea enviar la información al servidor, para esto es necesario poder acceder a la forma que hemos creado, en este caso accedemos mediante su id de la siguiente forma **`Ext.getCmp('formaCrearNodo')`**, y ahora podemos acceder a la forma recuperar los valores y enviar al servidor, es necesario realizar **`getForm()`** debido a que el panel que usamos "**`FormPanel`**" hereda de otra clase sus atributos, al realizar el **`submit`** o ingreso pueden existir 2 respuestas 1 correcta **`success`** u otra **`failure`**, básicamente lo que sucede es que cuando el servidor devuelve una respuesta sin excepciones del servidor como un error 500 significa que debe

ejecutarse el código en **success**, pero esto no significa de que todo el código se haya ejecutado correctamente, caso contrario como una respuesta 500, y no devuelva nada el servidor se irá a **failure** y presentará un mensaje de error, el código de **success** lo que hace es decodificar la respuesta, debido a que se recibe en formato de texto y es necesario convertirlo en un objeto JSON para que pueda ser utilizado. La ventana con el formulario luce así:



Para que el servido pueda procesar los datos enviados es necesario editar la clase árbol y aumentar el siguiente procedimiento:

```
function crear($params) {
       $texto = mysql_real_escape_string($params['nombre']);
       $padre = intval($params['padre']);
       $query = "INSERT INTO arbol (nombre, parent_id) VALUES ('$texto', $padre) ";
       if (mysql_query($query, $this->bd)) {
              $res = mysql_query("SELECT MAX( id ) AS id FROM arbol");
              $fila = mysql_fetch_assoc($res);
              $this->presentar(array('success' => true, 'id' => $fila['id']));
       }else{
              $this->presentar(array('success' => false));
              }
       }
```

Se recibe los parámetros nombre y padre y se inserta en la base de datos en caso de que se ejecute correctamente la inserción se retorna un **array** con **success => true**, el código de la función para editar es muy similar, solo que en lugar de enviar el id del padre se envía el id del nodo que se está editando su nombre y es necesario que la forma tenga como valor inicial el texto

```
function editarNodo() {
var nodo = Ext.getCmp('arbol').getSelectionModel().selNode;
   (new Ext.Window({
        title: 'Editar Nodo', width: 250, modal: true,
        items: [{
        xtype: 'form', id: 'formaEditarNodo', labelWidth: 40, url: 'arbol.php',
        items:[{
        xtype: 'hidden', name: 'accion', id: 'accion', value: 'editar'
        },{
        xtype: 'hidden', name: 'id', id: nodo.id, value: nodo.id
        },{
        xtype: 'textfield', fieldLabel: 'Texto', name: 'nombre', id: 'texto',
allowBlank: false,
        value: nodo.attributes.text // Se le da el valor del nodo seleccionado
        }]
    }],
       …
 success: function() {
        forma. findParentByType('window').hide();
        nodo.setText(nombre);
     },
…..
```

Quarer ser libre es ser libre

Desarrollo

Querer ser libre es ser libre

Desarrollo

Como ven solo muestro la parte que ha sido modificada y que no es similar al de la función **crearNodo**, lo que se hace es crear un formulario con tres campos y hay que darle como valor inicial al campo de texto el nombre del nodo seleccionado y el código para editar en el servidor es el siguiente

```
function editar($params) {
        $id = intval($params['id']);
        $nombre = mysql_real_escape_string($params['nombre']);

        $query = "UPDATE arbol SET nombre='$nombre' WHERE id=$id";
        if (mysql_query($query, $this->bd)) {
                    $this->presentar(array('success' => true));
             }else{
                    $this->presentar(array('success' => false));
             }
    }
```

Básicamente se actualiza con un **query** el nodo que estamos editando y se envía la respuesta. Ahora debemos poder borrar los nodos, para eso debemos insertar este código:

```
function borrarNodo() {
// Inicio de llamada al servidor
var arbol = Ext.getCmp('arbol');
var nodo = arbol.getSelectionModel().selNode;
Ext.Ajax.request({
    url:'arbol.php',
    method: 'POST',
    params: {accion: 'borrar', id: nodo.id},
    success: function(resp, o) {
    // Capturar excepcion
    try{
             // Decodifica las cadenas de texto y las convierte en JSON
             r = Ext.decode(resp.responseText);
             }catch(e) {
             o.failure();
        }
    // Verificar respuesta exitosa
    if(!r.success) {
          // Se muestra un mensaje de error recibido desde el servidor
          Ext.MessageBox.show({
          title: 'Error',
          msg: r.error,
          buttons: Ext.MessageBox.OK,
          icon: Ext.MessageBox.ERROR
          });
          }else {
               nodo.remove();
          }
               arbol.enable(); //Habilitar Panel

          return false;
          },
      //Funcion de falla
      failure: function(resp, o) {
      arbol.enable();
```

```
      // Se muestra un mensaje de error
      Ext.MessageBox.show({
         title: 'Error',
         msg: 'Existion un error al borrar el nodo!',
         buttons: Ext.MessageBox.OK,
         icon: Ext.MessageBox.ERROR
      });
    }
  });
}
```

En caso de borrar un nodo es necesario realizar una llamada Ajax, primero se obtiene el id del nodo que está seleccionado y el cual será borrado; se envían los datos al servidor y si éste da una respuesta afirmativa se elimina el nodo del árbol, **`nodo.remove()`**, para poder procesar esta llamada Ajax en el servidor es necesario este código:

```
function borrar($params) {
$id = intval($params['id']);
$res = mysql_query("SELECT COUNT(*) AS total FROM arbol WHERE parent_id=$id", $this-
>bd);
$fila = mysql_fetch_assoc($res);
if($fila['total'] <= 0) {
    $query = "DELETE FROM arbol WHERE id=$id";
    mysql_query($query, $this->bd) or die("Error en la base de datos");
    $this->presentar(array('success' => true) );
}else{
    $this->presentar(array('success' => false, 'error' => 'Debe seleccionar un nodo sin
hijos') );
       }
}
```

En caso del servidor lo que hace es recibir los parámetros, se busca mediante el **`id`**, si el nodo que se está borrando tiene hijos, no es posible borrar, caso contrario se elimina de la base de datos y se envía una respuesta afirmativa al navegador.

Referencias

[1] http://extjs.com

[2] http://getfirebug.com

Autor



Boris Barroso
boriscyber@gmail.com

Desarrollo Ágil con Ruby on Rails (3ra Parte)

Ruby on Rails es un framework de desarrollo web ágil, elaborado por David Heinemeier Hansson, que el 2004 lanzó la versión pública a partir de desarrollar el proyecto Basecamp, Ruby on Rails (RoR) está desarrollado en el lenguaje de programación Ruby, RoR actualmente se encuentra en la versión 2.1.1 en la que contribuyeron más de 1400 desarrolladores del mundo con 1600 parches al framework, por estos datos RoR es uno de los proyectos open source con más movimiento actual.

Introducción

Imaginemos que estamos dentro de la oficina o el lugar de trabajo, llega un cliente queriendo un proyecto X, tenemos un listado de requerimientos, los cuales son bastantes, la mitad de ellos lo conocemos y sabemos como hacerlo pero la otra mitad no, y como siempre nos piden un precio por este proyecto (muy bajo por cierto), y para quedar bien con el cliente debemos de darle un margen pequeño de tiempo de desarrollo, entonces tenemos que darle los mejores precios en el tiempo mínimo y un producto bueno, creo que con magia lo lograríamos, pero en la vida real tendremos que sacrificar uno o dos de las tres cualidades que mencionamos.

Una de las grandes características dentro del desarrollo ágil y algo que se encuentra dentro del Framework rails, es la reutilización extensiva de código, que dentro de Rails se denominan Plugins y Gemas, Estos son un conjunto de funciones comunes para un problema recurrente, esta funcionalidad está empaquetada en namespace, muy fácil de agregar a nuestros proyectos.

La diferencia principal entre Plugins y Gemas es que un plugin debe de instalarse para cada proyecto que necesitemos, mientras una gema es el paquete comprimido que requiere instalarse en la máquina (servidor de desarrollo) una vez, para utilizarla en todos los proyectos que necesiten esa funcionalidad.

A continuación de nuestro tutorial, requerimos que se paginen los resultados de las revistas también que se cree un entorno de pruebas real que tenga muchos usuarios que tengan escritos bastantes artículos y muchas revistas publicadas.

Para esto instalaremos tres gemas que son: faker que crea datos ficticios para nombres, direcciones, etc; populator crea datos ficticios para párrafos, líneas y números randomicos y por último will_paginate, que permite paginar de forma fácil nuestros resultados.

Instalaremos las siguientes gemas dentro de nuestra máquina:

Agregamos el repositorio de GitHub

```
gem sources -a http://gems.github.com
```

instalamos las gemas:

```
[sudo] gem install mislav-will_paginate
[sudo] gem install populator
[sudo] gem install faker
```

Ahora llenaremos nuestro proyecto con digamos 100 usuarios registrados y cada usuario escribió entre 1 a 4 artículos publicados dentro de 50 revistas publicadas.

Creamos el archivo **populator.rake** dentro de **app/libs/rake/**

```
namespace :db do
  desc "Llenar la base de datos"
  task :populate => :environment do
    require 'populator'
    require 'faker'

    [Autor, Articulo, Revista].each(&:delete_all)

    Autor.populate 100 do |p|
      p.nombre = Faker::Name.first_name
      p.apellidos = Faker::Name.last_name
      p.email = Faker::Internet.email(p.nombre)
      p.telefonos = Faker::PhoneNumber.phone_number
      p.biografia = Populator.sentences(2..10)

      Articulo.populate 1..4 do |a|
        a.titulo = Populator.words(5..10)
        a.contenido = Populator.sentences(2..10)
        a.revista_id = 1..50
        a.autor_id = p.id
      end
    end

    i=1
    Revista.populate 50 do |r|
      r.nombre = Populator.words(5..10)
      r.fecha_publicacion = 1.year.ago .. Time.now
      r.numero = i
      r.editorial = Populator.sentences(5..10)
      i +=1
    end
  end
end
```

Ahora hacemos correr nuestra tarea con el comando **rake db:populate** y llenará la base de datos con datos de demostración, el código es muy simple de entender.



Ahora utilizaremos la gema para paginar los resultados, ésta se llama **will_paginate**, un componente muy simple de utilizar, ahora hacemos que la aplicación cargue la librería al momento de inicializarse el proyecto, para esto modificamos el archivo:

config/envinroment.rb

```
config.gem 'mislav-will_paginate', :lib => 'will_paginate'
```

Quedando así la archivo

config/envinroment.rb

```
RAILS_GEM_VERSION = '2.2.2' unless defined? RAILS_GEM_VERSION
require File.join(File.dirname(__FILE__), 'boot')
Rails::Initializer.run do |config|
  config.time_zone = 'UTC'
  config.gem 'mislav-will_paginate', :lib => 'will_paginate'
  config.action_controller.session = {
    :session_key => '_atix_session',
                               :secret                                              =>
'69df0393a111d39a7f5d007a0400f244680076ffbdc572b7e519f2472608e4a40ea2fa51fe651dda7f4ed1
4bf8111f25879ea3096c957b828ff0095349db2901'
  }
end
```

Bien con esto ya tenemos configurado la gema, ahora vayamos al controlador de las revistas ubicado en **app/controllers/revistas_controller.rb** y modificamos la función **index** quedando así:

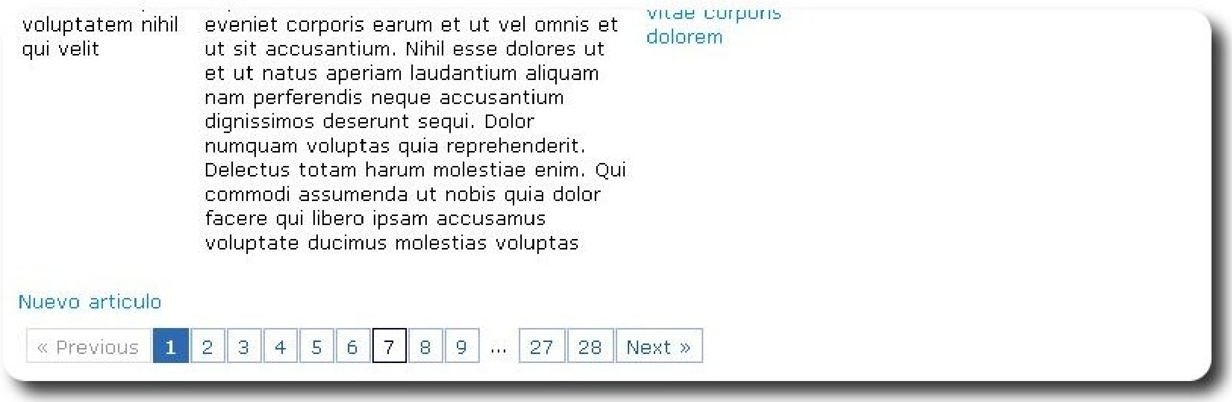
```
  def index
    @revistas = Revista.paginate(:page => params[:page])
    respond_to do |format|
      format.html # index.html.erb
      format.xml  { render :xml => @revistas }
    end
  end
```

Ahora vamos a la vista de esta acción **app/views/index.html.erb** y añadimos el siguiente **helper: will_paginate**, mostrado a continuación:

```
<h1>Listado de revistas</h1>
<table>
  <tr>
    <th>Nombre</th>
    <th>Fecha publicacion</th>
    <th>Numero</th>
    <th>Editorial</th>
    <th>Publicado</th>
    <th colspan="4">Acciones</th>
  </tr>
<% for revista in @revistas %>
  <tr>
    <td><%=h revista.nombre %></td>
    <td><%=h revista.fecha_publicacion %></td>
    <td><%=h revista.numero %></td>
    <td><%=h truncate(revista.editorial, :length => 100) %></td>
    <td><%=h revista.created_at.strftime("%Y/%m/%d") %></td>
    <td><%= link_to 'Articulos', revista_articulos_path(revista) %></td>
    <td><%= link_to 'Mostrar', revista %></td>
    <td><%= link_to 'Editar', edit_revista_path(revista) %></td>
     <td><%= link_to 'Borrar', revista, :confirm => 'Esta seguro?', :method => :delete
%></td>
  </tr>
<% end %>
</table>
<%= will_paginate @revistas %>
<br />
<%= link_to 'Nuevo revista', new_revista_path %>
```

Con esto ya tenemos nuestros resultados paginados, por defecto agarra 30 registros como límite por página esto lo podemos modificar con el atributo :**per_page** digamos que queremos que muestre 10 registros a la vez, esto en el controlador:

```
@revistas = Revista.paginate(:page => params[:page], :per_page => 10)
```



También podemos utilizar toda la magia que nos trae **active_record**, dentro de la consulta, como ejemplo si queremos todos los artículos escritos hace 1 mes y ordenados por la fecha de publicación, paginados por 10 registros tenemos:

```
@revistas = Revista.paginate(:page => params[:page], :conditions => ["fecha_publicacion > ?", 1.month.ago], :order => 'fecha_publicacion')
```

Otra de las características mágicas es que también realiza la paginación a consultas **sql** personalizadas, como por ejemplo: Si queremos todos los autores de una revista n, mostrando su el título de su publicación y el nombre de la revista, mostrando los resultados por cada 10 registros.

Para este requerimiento tendríamos la siguiente consulta **sql**, imaginemos que es para la revista 1:

```
SELECT
  revistas.nombre,
  revistas.fecha_publicacion,
  articulos.titulo,
  autores.nombre,
  autores.apellidos
FROM
  articulos
  INNER JOIN autores ON (articulos.autor_id = autores.id)
  INNER JOIN revistas ON (articulos.revista_id = revistas.id)
WHERE
  revistas.id = 1
```

Creamos el método **sql** dentro del controlador **app/controllers/revista_controller.rb** y escribimos:

```
  def sql
    sql = ["SELECT
              revistas.nombre as nombre_revista,
              revistas.fecha_publicacion,
              articulos.titulo as titulo_articulo,
              autores.nombre,
              autores.apellidos
            FROM
              articulos
              INNER JOIN autores ON (articulos.autor_id = autores.id)
              INNER JOIN revistas ON (articulos.revista_id = revistas.id)
            WHERE
              revistas.id = ?", params[:id]]
    @autores = Revista.paginate_by_sql(sql, :page => params[:page], :per_page => 10)
  end
```

La vista correspondiente en **app/views/sql.html.erb**

```
<h1>Listado de autores por revistas</h1>
<table>
  <tr>
    <th>Nombre</th>
    <th>Fecha publicacion</th>
    <th>Titulo articulo</th>
    <th>titulo revista</th>
  </tr>
<% for autor in @autores %>
  <tr>
    <td><%= "#{autor.nombre} #{autor.apellidos}"  %></td>
    <td><%=h autor.fecha_publicacion.strftime("%Y/%m/%d") %></td>
    <td><%=h autor.titulo_articulo %></td>
    <td><%=h autor.nombre_revista %></td>

  </tr>
<% end %>
</table>
<%= will_paginate @autores %>
```

Tendremos que modificar el archivo **config/routes.rb** para incorporar dentro de nuestras rutas el **sql_revista_path** para esto añadimos lo siguiente:

```
ActionController::Routing::Routes.draw do |map|
  map.resources :autores, :has_many => :articulos
  map.resources :articulos
  map.resources :revistas, :has_many => :articulos, :member => {:sql => :get}
  map.root :controller => "revistas"
  map.connect ':controller/:action/:id'
  map.connect ':controller/:action/:id.:format'
end
```

y por último un pequeño cambio en **app/views/revistas/index.html.erb**, en donde es:

```
<td><%=h revista.nombre %></td>
```

Cambiamos por:

```
<td><%= link_to(revista.nombre, sql_revista_path(revista)) %></td>
```

Conclusiones.

Dentro del desarrollo ágil, se pregona la reutilización de código para agilizar la realización de proyectos, para esto necesitamos de la utilización de plugins y/o gemas, el propio framework rails está desarrollado por módulos o gemas, lo que permite incrementar la modularidad dentro del framework.

Bien eso es todo por ahora, espero les sirva; repositorios de código con plugins y gemas se encuentran en **`rubyforge.org`** y **`hithub.com`**

Además el código fuente del proyecto se encuentra en: **`http://github.com/carakan/atix/tree/master`**

Referencias

[1] http://www.rubyonrails.org/

[2] http://www.rubyforge.org/

Autor



Carlos Ramos

Lic. Informática UMSA
Lider de Wiebia, soluciones web 2.0
carakan@gmail.com
Blog personal http://ww.carakan.com
http://www.wiebia.com

Full Text Search con PostgreSQL

Dentro la recuperación de texto integral, el full text serach, se refiere a la técnica de buscar un documento almacenado en una base de datos o en un ordenador, el motor de búsqueda analiza cada una de las palabras en cada documento almacenado, dando como resultado los documentos en los cuales esta presente cada una de las palabras buscadas.



Introducción

Uno de los aspectos decisivos para el éxito o fracaso de un proyecto de software es su sistema de búsquedas y la calidad de los resultados que provee al usuario final.

A lo largo de este artículo veremos las posibilidades que nos ofrece PostgreSQL para la implementación de un sistema de recuperación de información simple pero muy potente.

Preparando el terreno de juego

Para empezar cabe señalar que en este artículo utilizaremos la versión 8.3 de PostgreSQL, dado que a partir de ésta versión el módulo de Full Text Search se encuentra incluido dentro del core del motor de bases de datos, evitándonos el tener que incluir dichas funcionalidades haciendo uso de los contrib que vienen junto a la distribución principal de PostgreSQL.

Para empezar vamos a crear una estructura y unos usuarios que nos permitan trabajar correctamente.

Como fuente de datos utilizaremos unos cuantos artículos publicados en ATIX tomados al azar, empecemos.

Crearemos una base de datos y un usuario:

```
CREATE DATABASE FTS_prueba;
CREATE USER fts_user;
ALTER DATABASE FTS_prueba OWNER fts_user;
ALTER USER fts_user PASSWORD '*********';
```

Utilizaremos una tabla similar a ésta en la que se almacenarán los artículos publicados.

```
CREATE TABLE articulo
(
  id serial NOT NULL,
  texto text NOT NULL,
  titulo text NOT NULL,
  autor character varying(200) NOT NULL,
  texto_tsv tsvector NOT NULL,
  CONSTRAINT articulo_pkey PRIMARY KEY
(id)
)
```

La llenaremos con unos cuantos artículos para poder empezar a trabajar:

```
INSERT INTO articulo (texto, titulo,
autor, texto_tsv)
VALUES ('Por que darle una oportunidad?
Tuquito es una distribución GNU/Linux,
que implementa ...', 'Tuquito 2.0 Dale
una oportunidad',
'Oscar Choque Monzón',
to_tsvector('spanish', 'Por que darle una
oportunidad? Tuquito es una distribución
GNU/Linux, que implementa ...'));
```

La estructura de la tabla es bastante simple, cada artículo cuenta con un título, autor y cuerpo del artículo (texto), lo único especial y que vamos a aclarar a continuación es el campo texto_tsv, dicho campo del tipo tsvector almacena el cuerpo del artículo procesado utilizando la función to_tsvector(), dicha función es la encargada de transformar un texto a una estructura de datos adecuada para su procesamiento por el motor de bases de datos, la veremos más adelante en el artículo, este campo de nuestra estructura de artículos nos será útil más adelante.

Documentos y consultas

Un documento en el contexto de FTS en PostgreSQL es tan simple como una tupla de una tabla dentro una base de datos, una tupla que puede contener tantos campos como sea necesario, una consulta es simplemente un bloque de texto que se procesa para encontrar coincidencias dentro de un documento. Ambos componentes de nuestro sistema necesitan ser convertidos a estructuras especiales que permitan su procesamiento, los documentos se convertirán a la estructura tsvector y las consultas a la estructura tsquery.

Un tsvector es una estructura que contiene la información más relevante del documento que sirve para ser indexada. Todas las operaciones de búsqueda se realizan sobre los tsvector, los documentos originales solo son necesarios para devolver al usuario los resultados positivos de las búsquedas.

Un tsquery es una estructura que contiene los términos de búsqueda como lexemas normalizados, además de los operadores AND, OR y NOT que sean necesarios.

Procesando documentos

Procesar documentos para llevarlos a la estructura de un tsvector es tan sencillo como utilizar la función predefinida to_tsvector(), veamos un ejemplo, un fragmento del editorial del primer número de ATIX:

```
SELECT to_tsvector('spanish', 'Se dice
que un hombre debería hacer 3 cosas
importantes en su vida: tener un hijo,
plantar un árbol y escribir un libro; con
un poco de analogía podríamos comenzar a
contar la historia de la comunidad ATIX,
que empezó no teniendo un hijo sino
varios, hijos que durante este tiempo le
brindaron su tiempo y colaboración, poca
o mucha no interesa, lo importante es que
colaboraron');
```

Lo primero que notamos al utilizar la función es la utilización de un primer parámetro ‘spanish’, dicho parámetro, que utilizaremos a lo largo de todo el artículo, tiene el objetivo de indicarle al motor que para este procesamiento utilizaremos la configuración del idioma español (si, podemos utilizar muchos idiomas, pero también podemos definir un idioma por defecto en el archivo de configuración postgresql.conf), el segundo parámetro es simplemente el texto a procesar.

Como resultado obtenemos una salida así:

```
"'3':8 'cos':9 'deb':6 'dic':2 'hac':7
'hij':16,44,47 'poc':58 'sin':45 'ten':14
'vid':13 'atix':38 'cont':32 'libr':23
'much':60 'podr':29 'vari':46 'arbol':19
'brind':53 'comun':37 'empez':40
'hombr':5 'plant':17 'tiemp':51,55
'analog':28 'comenz':30 'escrib':21
'import':10,64 'colabor':57,67
'histori':34 'interes':62"
```

Al ver este resultado podemos notar varias cosas:

- ✔ Se han eliminado las stopwords (“Se”, “dice”, “que”, “un”, “en”, “y”, “a”, etc.) y los caracteres de puntuación.
- ✔ Se han normalizado todas las palabras y se las ha convertido a lexemas, almacenando cada lexema asociado a la posición de la(s) palabras en el texto (por ejemplo, “importantes” e “importante” se han unificado en el lexema “import” que esta asociado a las posiciones 10 y 64 respectivamente).
- ✔ Si dentro del documento existieran numerales, estos son transformados a cadenas y tratadas de la misma forma

que cualquier otra palabra, obviamente no pueden ser convertidas a su forma primitiva (lexemas).

Procesando consultas

Para llevar una consulta al formato tsquery simplemente utilizamos la función to_tsquery() ó plainto_tsquery(), tomemos como ejemplo la siguiente consulta:

```
SELECT to_tsquery('spanish', 'Programando
& con & Python');
Obtenemos el resultado:
       "'program' & 'python'"
```

Se ha procesado la consulta eliminando las stopwords y convirtiendo los demás términos a lexemas. A diferencia de un tsvector, los términos no se han asociado con ninguna posición sino que se los ha relacionado con operadores lógicos, por defecto el operador AND (&).

Si no queremos preprocesar la consultas (separarlas por palabras y juntarlas con operadores lógicos) antes de convertirlas al formato tsquery, podemos utilizar la función plainto_tsquery() que hace ese trabajo por nosotros:

```
SELECT plainto_tsquery('spanish',
'Programando con Python');
El resultado:
       "'program' & 'python'"
```

La diferencia radica en los operadores que podemos utilizar al momento de definir la consulta, con to_tsquery() podemos utilizar los operadores lógicos AND. OR y NOT, con plainto_tsquery() se utiliza el operador AND únicamente.

Realizando búsquedas

Ya sabemos como procesar los documentos y como procesar y definir los términos de búsqueda, ahora veremos como efectuar una búsqueda sobre una tabla.

```
SELECT titulo
FROM articulo
WHERE to_tsvector('spanish', texto) @@
to_tsquery('framework & php');
```

Realizamos la búsqueda sobre la columna texto de nuestra tabla artículo y nos devolverá el(los) título(s) del(los) artículo(s) en caso de que la búsqueda sea exitosa:

```
"Symfony Framework para el desarrollo de
aplicaciones web"
```

Si deseáramos realizar la búsqueda no solo sobre el cuerpo del artículo sino también sobre el título, podemos modificar la consulta de este modo:

```
SELECT titulo
FROM articulo
WHERE to_tsvector('spanish', titulo ||
texto) @@ to_tsquery('framework & php');
```

Creando y usando índices

Dependiendo del tamaño de la base de datos, una búsqueda determinada puede tornarse lenta dependiendo de la cantidad de documentos que deben ser procesados, no es lo mismo realizar una búsqueda en 10 documentos que hacerla sobre 1000 o más.

Con el objetivo de no realizar las búsquedas recorriendo y procesando todos los documentos cada vez, se tiene la posibilidad de definir índices para acelerar la obtención de los resultados.

Para nuestra tabla usaremos los índices especiales tipo GIN (Generalized Inverted Index):

```
CREATE INDEX articulo_idx ON articulo
USING gin(to_tsvector('spanish', texto));
```

Si deseamos definir un índice que agrupe a más de una columna de nuestra tabla:

```
CREATE INDEX articulo_idx ON articulo
USING gin(to_tsvector('spanish', titulo
|| texto));
```

Resaltando resultados

Una característica interesante de los sistemas de búsqueda es la presentación que se hace de los resultados resaltando la parte del documento que coincide con el criterio de búsqueda.

Cuando efectuamos una búsqueda en este tipo de sistemas podemos visualizar los fragmentos del documento con un color diferente al del resto del texto.

Si queremos implementar esta característica en PostgreSQL y Full Text Search debemos hacer uso de la función ts_headline() que nos facilita los resultados de una consulta con una variedad de opciones que veremos a continuación.

```
SELECT ts_headline('spanish', 'Se dice
que un hombre debería hacer 3 cosas
importantes en su vida: tener un hijo,
plantar un árbol y escribir un libro; con
un poco de analogía podríamos comenzar a
contar la historia de la comunidad ATIX,
que empezó no teniendo un hijo sino
varios, hijos que durante este tiempo le
brindaron su tiempo y colaboración, poca
o mucha no interesa, lo importante es que
colaboraron', to_tsquery('ATIX'));
```

El resultado por defecto es este:

```
"<b>ATIX</b>, que empezó no teniendo un
hijo sino varios, hijos que durante este
tiempo le brindaron"
```

Obtenemos una parte pequeña del documento con la(s) partes que coinciden con el criterio de búsqueda encerradas entre tags HTML <b>, insertar luego este resultado en una plantilla HTML es bastante trivial. Si deseamos definir otro tipo de resaltado de los resultados podemos hacer uso de las opciones StarSel y StopSel de la siguiente forma:

```
SELECT ts_headline('spanish', 'Se dice
que un hombre debería hacer 3 cosas
importantes en su vida: tener un hijo,
plantar un árbol y escribir un libro; con
un poco de analogía podríamos comenzar a
contar la historia de la comunidad ATIX,
que empezó no teniendo un hijo sino
varios, hijos que durante este tiempo le
brindaron su tiempo y colaboración, poca
o mucha no interesa, lo importante es que
colaboraron', to_tsquery('ATIX'),
'StartSel = <i>, StopSel = </i>');
```

Devolviéndonos este resultado:

```
"<i>ATIX</i>, que empezó no teniendo un
hijo sino varios, hijos que durante este
tiempo le brindaron"
```

Si queremos definir el número de palabras que deberá tener nuestro resultado podemos hacer uso de las opciones MaxWords y MinWords así:

```
SELECT ts_headline('spanish', 'Se dice
que un hombre debería hacer 3 cosas
importantes en su vida: tener un hijo,
plantar un árbol y escribir un libro; con
un poco de analogía podríamos comenzar a
contar la historia de la comunidad ATIX,
que empezó no teniendo un hijo sino
varios, hijos que durante este tiempo le
brindaron su tiempo y colaboración, poca
o mucha no interesa, lo importante es que
colaboraron', to_tsquery('ATIX'),
'StartSel = <i>, StopSel = </i>, MinWords
= 10, MaxWords = 20');
```

Ranking de resultados

Para finalizar este artículo veremos una última característica interesante, la del ranking de resultados. Comúnmente podemos ver que cuando utilizamos el buscador de un sistema de información obtenemos los resultados ordenados por algún criterio definido por su programador, ya sea por la fecha de su publicación (los documentos más recientes deberían ser más importantes) o por la posición de los documentos en una tabla en la base de datos (los documentos insertados inicialmente serán los que se muestren primero).

Con PostgreSQL también tenemos la opción de devolver al usuario los resultados de sus búsquedas ordenados por relevancia, tomando unos criterios como estos:

- ✔ Cuantas coincidencias tiene una búsqueda en un determinado documento.
- ✔ Que tan cercanas están esas coincidencias en dicho documento.
- ✔ En que parte del documento esta la coincidencia (si está al principio del documento debería ser más relevante)

Con estos criterios, entre otros más, se les asignan pesos a los resultados y conforme a dichos pesos se mostrarán primero los resultados con mayor relevancia.

Para hacer uso de esta característica debemos utilizar la función ts_rank() de la siguiente forma:

```
SELECT titulo, autor, ts_rank(texto_tsv,
to_tsquery('spanish', 'Java')) AS ranking
FROM articulo
ORDER BY ranking DESC;
```

Cabe hacer notar que la ejecución de la función ts_rank() se la debe realizar pasándole como parámetro, un atributo del tipo tsvector, al contrario que las anteriores funciones que podíamos pasarles textos simples, en éste caso el parámetro texto_tsv es del tipo tsvector por lo que no tenemos problema alguno.

A continuación podemos ver los resultados de una búsqueda utilizando la consulta “Java”, se muestran todos los artículos con su respectivo ranking, en caso de tener un ranking de cero significa que la consulta es negativa para ese documento:

Output pane

Data Output | Explain | Messages | History

	titulo text	autor character varying(200)	ranking real
1	JasperReport + Ireport Generación de Reportes en Java	Cristhian Choquecallata Machicado	0.0889769
2	Ogre3D Una alternativa para el desarrollo gráfico	Arnold Marcelo Guzmán	0.0607927
3	Vim guía de Supervivencia (2da parte)	Williams Israel Chorolque Choque	0
4	VSFTPD: una forma de permitir y mantener control de la transferencia de a	Rocio Figueroa	0
5	Openssh la magia de la administración remota	Lucy Ayarde Romero	0
6	Bazaar la herramienta para el control de versiones de forma distribuida	Esteban Saavedra López	0
7	ReStructuredText: Realizando documentos de forma rápida y sencilla	Esteban Saavedra López	0
8	Blender: Una herramienta 3D libre	Arnold Marcelo Guzmán	0
9	Compartiendo archivos pormedio de Apache	Lucy Ayarde Romero	0
10	Amanda: una forma sencilla de implementar un sistema de respaldos	Lucy Ayarde Romero, Esteban Saavedra López	0
11	GNU Privacy Guard Intercambiando mensajes y documentos de forma segu	Esteban Saavedra López	0
12	Tuquito 2.0 Dale una oportunidad	Oscar Choque Monzón	0
13	Interactuando con GNU Privacy Guard	Esteban Saavedra López, Joseph Sandoval Falomici	0
14	Symfony Framework para el desarrollo de aplicaciones web	Esteban Saavedra López	0

Conclusiones

La utilización de PostgreSQL y su módulo de Full Text Search nos da dos grandes beneficios al momento de desarrollar las funcionalidades de búsqueda en nuestros sistemas, por un lado aprovechamos muchas características de los sistemas de recuperación de información que ya están implementadas en el motor de base de datos y por otro lado, centralizamos el componente de búsquedas de nuestro sistema en el mismo servidor.

Espero que a lo largo de este corto artículo haya podido mostrar que si elegimos PostgreSQL, el camino hacia un sistema de búsquedas efectivo se hará más divertido e interesante.

Referencias

[1] http://www.postgresql.org

[2] http://www.postgresql.org/docs/8.3/static/textsearch.html

Autor



Rodrigo Soliz Rocabado

rodrifer@gmail.com

Empaquetando nuestras aplicaciones para distribuirlas

Hoy en día son muchas las personas que se dedican al desarrollo de software, pudiendo ser el desarrollo de aplicaciones completas tanto para escritorio como para Web, o de simples aditamentos o extenciones a una ya desarrollada; éstas personas se ven en la imperiosa necesidad de contar con un medio que les permita distribuir éstas aplicaciones, dentro sus entornos, o hacerlos públicos en Internet para beneficio o conocimiento de otros usuarios.

El empaquetar una aplicación representa una de las mejores y más recomendadas formas de distribuir una aplicación a usuarios de cierto entorno, o ponerla a disposición en algún repositorio disponible en Internet.



Introducción

Muchos cuando empezamos a desarrollar nuestras primeras aplicaciones, para ponerlas en producción o a disposición de otros usuarios, generalmente nos limitamos a realizar una simple copia de los programas y archivos en las máquinas destino; representando todo ésto un proceso laborioso y muchas veces complicado, ya que representa que en muchos casos tengamos que instalar ciertos programas o librerías que hacen falta para que nuestra aplicación funcione correctamente y más aún cuando el software instalado en cada PC varia de una a otra.

Para evitar todo este proceso tedioso, es recomendable empaquetar nuestras aplicaciones, para que el proceso de distribución sea transparente para los usuarios destino, y sobre todo que este proceso tenga la facilidad de realizarse de forma automática, considerando además sus posibles actualizaciones, parchado de bugs, mantención y acceso a los desarrolladores de la misma.

El proceso de empaquetado no es un proceso tan complicado como parece, pero tiene ciertas variantes dependiendo principalmente de la distribución a la cual esta destinada la aplicación. Los paquetes creados consideran básicamente cierta información común (las fuentes, la arquitectura, la descripción, las dependencias, los requisitos, etc), ésta información es contemplada en uno o dos archivos de especificación y/o control en ambos casos; archivos que son la principal fuente de descripción y especificación para futuras regeneraciones del paquete en si.

Una de las mayores facilidades que provee el empaquetado de aplicaciones es: su fácil distribución, actualización y parchado de bugs en algunos casos, tareas, que a la postre se realizan de forma automática, con tan solo poner a disposición los nuevos paquetes o parches según corresponda.

Elementos de un proceso de empaquetado

Son varios los elementos que se deben considerar al momento de empaquetar una

aplicación, entre los más comunes podemos denotar:

- La arquitectura destino del paquete
- El empaquetado de fuentes o binarios
- La distribución destino
- Información del paquete, expresada mediante directivas (descripción, dependencias, etc.)
- Firma digital del paquete, para garantizar la autenticidad del mismo

En su generalidad esta información es contemplada en los archivos destinados a especificar y/o controlar el proceso de empaquetado en si.

Formatos de empaquetado y distribución

Por lo general cada distribución tiene su propia forma de empaquetar sus aplicaciones; pero hoy en día existen dos grandes familias que han impuesto supremacía en este tema como son:

- **RMP** (Redhat Package Manager) de la familia de RedHat (RHEL, Fedora, CentOS), Mandriva, Suse
- **Deb** de la familia de Debian (Debian, Ubuntu, y derivados).

RPM

RPM es el gestor de paquetes de Red Hat (Red Hat Package Manager). Aunque aparece Red Hat en su nombre, la intención es que sea un sistema de empaquetado abierto y disponible para el uso de cualquiera. Permite a los usuarios tomar el código fuente (source code) y empaquetarlo en forma de fuentes y binaria de forma que los ficheros binarios sean fácilmente instalables y rastreables y las fuentes puedan ser reconstruidas con facilidad. También gestiona una base de datos de todos los paquetes y sus ficheros que puede ser usada para verificar paquetes e interrogarla para obtener información acerca de ficheros y/o paquetes.

Características

- RPM está diseñado para disponer de potentes parámetros de consulta. Usted puede hacer búsquedas de paquetes a lo largo de toda la base de datos o sólo de ciertos ficheros. También puede encontrar fácilmente a qué paquete pertenece un fichero y de dónde proviene. Los ficheros RPM en sí mismos son archivos comprimidos, pero puede consultar paquetes independientes fácil y rápidamente, gracias a una cabecera binaria a medida añadida al paquete con toda la información que puede necesitar, almacenada sin comprimir. Esto permite consultas rápidas.
- Otra poderosa característica es la habilidad de verificar paquetes. Si está preocupado por haber borrado algún fichero importante, sólo tiene que verificar el paquete. Quedará cumplidamente informado de cualquier anomalía. Si ésto ocurriera, podrá reinstalar el paquete si lo considera necesario. Cualquier fichero de configuración que usted tenga quedará a salvo.

Requisitos

Al momento de empaquetar una aplicación, se precisa tener instalados los siguientes paquetes:

- rpm-build
- rpm-devel
- gnupg
- install
- gcc*
- automake*
- autoconf*

Todos y cada uno de éstos pueden ser instalados mediante el gestor de paquetes de la distribución utilizada.

Creando nuestro paquete

Por tratarse de un artículo que pretende demostrar el empaquetamiento de aplicaciones, su procedimiento y características del mismo y para que se tenga una mejor idea de lo que deseamos hacer, explicaremos el objetivo y las características de nuestro paquete, bajo el siguiente detalle:

- ✔ Crearemos un paquete que incluya los distintos números de la revista Atix.
- ✔ El paquete dispondrá de actualizaciones, ésto representa que cada vez que se libere un nuevo número de la revista, también se liberara una actualización del paquete, que obviamente permita contar con el nuevo número de la revista Atix
- ✔ El paquete dispondrá de una firma digital, para garantizar la autenticidad del paquete
- ✔ El formato del paquete será un RPM, para distribuciones como RHEL, CentOS, Fedora, etc. (en posteriores entregas realizaremos el empaquetado para otras distribuciones)
- ✔ Una vez instalado el paquete, éste permitirá incluir los distintos números de la revista Atix en el menú de aplicaciones/educación, lo que facilitará el acceso directo a cada uno de los números de la revista

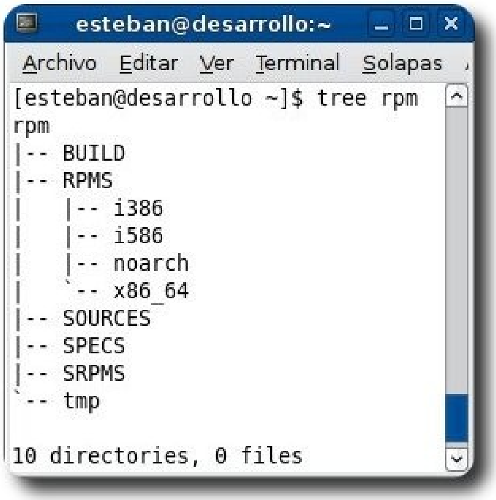
Detalles de la creación de paquetes

A continuacón detallamos la creación de paquetes.

Estructura de directorios

Para construir paquetes, el primer elemento importante es crear una estructura de directorios adecuada a tal efecto, de tal forma que esta estructura nos permita mantener orden y control sobre todos y cada uno de los archivos necesarios para la creación del paquete, así mismo de los archivos generados.

La estructura recomendada es la siguiente:



- ✔ **~/rpm/BUILD**: El directorio donde los códigos fuente se construyen.
- ✔ **~/rpm/RPMS**: Contiene los directorios, uno por cada arquitectura, que recibirán posteriormente los ficheros binarios compilados.
- ✔ **~/rpm/RPMS/i586**: El directorio donde se almacenarán los paquetes rpm para procesadores i586.
- ✔ **~/rpm/RPMS/x86_64**: El directorio donde se almacenarán los paquetes rpm para procesadores AMD64.
- ✔ **~/rpm/RPMS/noarch**: El directorio donde se almacenarán los paquetes rpm que sean independientes de la arquitectura del procesador.
- ✔ **~/rpm/SOURCES**: Los ficheros de código fuente (mipaquete.tar.bz2, por ejemplo).
- ✔ **~/rpm/SPECS**: Los ficheros de especificaciones que tendremos que construir.
- ✔ **~/rpm/SRPMS**: El rpm de código fuente tras su construcción.
- ✔ **~/rpm/tmp**: Para cosas temporales que rpm creará cuando construya sus paquetes.

Archivo .rpmmacros

Un segundo elemento importante dentro la creación de paquetes es el archivo **.rpmmacros**, que contiene una serie de parámetros de configuración que serán considerados al momento de crear el paquete. En caso de no existir este archivo se tomará lo parámetros y ubicaciones por defecto de la distribución

El contenido básico de este archivo, podría ser el que mostramos a continuación

~/.rpmmacros

```
# Descripcion del Path del paquete
%_topdir                %(echo $HOME)/rpm
%_tmppath               %(echo $HOME)/rpm/tmp
# Para que nuestros paquetes sean automáticamente firmados con GPG, incluimos las
siguientes lineas:
%_signature             gpg
%_gpg_name              RevistaAtix
%_gpg_path              ~/.gnupg
# De todas formas el paquete puede ser firmado despues de ser creado, haciendo uso de
la opción rpm --resign

# Detalles del empaquetador, la distribución y el vendedor.
%packager               Esteban Saavedra Lopez <estebansaavedra@yahoo.com>
%distribution           RHEL, CentOS, Fedora, Mandriva, Suse
%vendor                 Revista Atix <http://atix.opentelematics.org>

# Para incluir un subfijo propio en el nombre del paquete
%distsuffix             revista
```

Firma digital

Un tercer elemento importante es la firma digital, con la que firmaremos nuestros paquetes, proveyéndoles la autenticidad correspondiente.

Generalmente cuando instalamos un paquete, que no provenga de los repositorios oficiales, el sistema operativo nos alerta que estamos instalando un paquete del cual desconocemos su procedencia y corremos el riesgo de instalar software malicioso en nuestro sistema, por tal razón es conveniente instalar paquetes que dispondrán de una firma digital que garantice su autenticidad.

En el número 4 de la revista Atix, publique un artículo de como crear una llave pública y privada que sirve para firmar digitalmente cualquier documento, sugiero revisar ese artículo en caso de no conocer como se generan llaves para la firma de documentos.

Para nuestro ejemplo asumimos que tenemos creado una llave GPG llamada RevistaAtix, con la cual se firmarán nuestros paquetes.

Archivo de especificación

El cuarto elemento importante, el el archivo de especificación, que contiene una serie de opciones parametrizadas que permitirán caracterizar el paquete. Este archivo dispone de varias secciones como ser:

- ✔ **cabecera**:contiene un conjunto de campos que permiten identificar al paquete
- ✔ **description**: contiene una descripción detallada del paquete
- ✔ **%prep**:contiene el nombre de los scripts necesarios para desempaquetar y parchar las fuentes
- ✔ **%build**:contiene los comandos para construir y compilar los datos ya instalados en el directorio correcto
- ✔ **%install**:contiene los comandos necesarios para para realizar la instalación
- ✔ **%files**:contiene el detalle de los archivos componentes del paquete.
- ✔ **%clean**: contiene las instrucciones

que permiten limpiar los archivos temporales generados

- **%config**: contiene el listado de los ficheros de configuración de un paquete
- **%doc**:Contiene el listado de archivos de documentación del paquete
- **%changelog**: contiene el detalle de los diversos cambios hechos al paquete.
- **%pre**: guiones preinstalación
- **%post**: guiones postinstalación
- **%preun**: guiones predesistalación
- **%postun**: guiones postdesistalación

Debemos tener presente que no es necesario incluir todas las secciones, sino solo las que se consideren necesarias en cada caso, así como veremos en el ejemplo que desarrollamos.

Para una mejor comprensión del contenido de este archivo de especificación presentamos el desarrollado para nuestro ejemplo, y la explicación del mismo.

Cabecera

Contiene la información básica del paquete (nombre, versión, licencia, requerimiento, ...)

atix.spec

```
Name: atix
Version: 1.0.1
Release: 1.%{distsuffix}
URL: http://atix.opentelematics.org/
Summary: Numeros de la Revista Atix
License: GPL
Group: Documentation
Source0: atix-1.0.1.tar.bz2
BuildRoot: %{_tmppath}/%{name}-%{version}-buildroot
Requires: evince

%description
atix es un paquete creado por Esteban Saavedra Lopez <jesaavedra@opentelematics.org>.
El objetivo de este paquete es instalar los distintos numeros publicados de la Revista
de Software Libre Atix, y peridodicamente poseer actualizaciones que vayan incluyendo
los distintos numeros segun se vayan liberando.

Más información en: <http://atix.opentelematics.org>
```

En nuestro ejemplo el archivo atix-1.0.1.tar.bz2 , contiene todos los archivos del paquete (archivos .pdf de la revista, archivos .xpm de iconos, archivos .desktop de entradas al menú y los archivos de documentación)

%prep

Contiene el nombre de los scripts necesarios para desempaquetar y parchar las fuentes . En nuestro caso realizará la descompresión de los archivos fuente (source 0)

```
%prep
%setup -q -a 0
```

%install

Contiene los comandos necesarios para realizar la instalación ; incluyendo la creación de directorios en caso de no existir (-D), los atributos de los archivos (-m) y las ubicaciones donde se instalarán/copiarán los archivos contenidos en el paquete.

atix.spec (continuación)

```
%install
rm -rf $RPM_BUILD_ROOT
install -D -m 644 atix01.pdf $RPM_BUILD_ROOT/usr/share/RevistaAtix/atix01.pdf
install -D -m 644 atix02.pdf $RPM_BUILD_ROOT/usr/share/RevistaAtix/atix02.pdf
install -D -m 644 atix03.pdf $RPM_BUILD_ROOT/usr/share/RevistaAtix/atix03.pdf
install -D -m 644 atix04.pdf $RPM_BUILD_ROOT/usr/share/RevistaAtix/atix04.pdf
install -D -m 644 atix05.pdf $RPM_BUILD_ROOT/usr/share/RevistaAtix/atix05.pdf
install -D -m 644 atix06.pdf $RPM_BUILD_ROOT/usr/share/RevistaAtix/atix06.pdf
install -D -m 644 LEAME $RPM_BUILD_ROOT/usr/share/RevistaAtix/LEAME
install -D -m 644 TODO $RPM_BUILD_ROOT/usr/share/RevistaAtix/TODO
install -D -m 644 COPYING $RPM_BUILD_ROOT/usr/share/RevistaAtix/COPYING
install -D -m 644 ChangeLog $RPM_BUILD_ROOT/usr/share/RevistaAtix/ChangeLog
install -D -m 644 atix01.desktop $RPM_BUILD_ROOT/usr/share/applications/atix01.desktop
install -D -m 644 atix02.desktop $RPM_BUILD_ROOT/usr/share/applications/atix02.desktop
install -D -m 644 atix03.desktop $RPM_BUILD_ROOT/usr/share/applications/atix03.desktop
install -D -m 644 atix04.desktop $RPM_BUILD_ROOT/usr/share/applications/atix04.desktop
install -D -m 644 atix05.desktop $RPM_BUILD_ROOT/usr/share/applications/atix05.desktop
install -D -m 644 atix06.desktop $RPM_BUILD_ROOT/usr/share/applications/atix06.desktop
install -D -m 644 atix01.xpm $RPM_BUILD_ROOT/usr/share/pixmaps/atix01.xpm
install -D -m 644 atix02.xpm $RPM_BUILD_ROOT/usr/share/pixmaps/atix02.xpm
install -D -m 644 atix03.xpm $RPM_BUILD_ROOT/usr/share/pixmaps/atix03.xpm
install -D -m 644 atix04.xpm $RPM_BUILD_ROOT/usr/share/pixmaps/atix04.xpm
install -D -m 644 atix05.xpm $RPM_BUILD_ROOT/usr/share/pixmaps/atix05.xpm
install -D -m 644 atix06.xpm $RPM_BUILD_ROOT/usr/share/pixmaps/atix06.xpm
```

%clean

Contiene las instrucciones que permiten limpiar los archivos temporales generados

atix.spec (continuación)

```
%clean
rm -rf $RPM_BUILD_ROOT
```

%files

Contiene el detalle (atributos, ubicación) de todos los archivos componentes del paquete.

atix.spec (continuación)

```
%files
%defattr(0644,root,root)
/usr/share/RevistaAtix/atix01.pdf
/usr/share/RevistaAtix/atix02.pdf
/usr/share/RevistaAtix/atix03.pdf
/usr/share/RevistaAtix/atix04.pdf
/usr/share/RevistaAtix/atix05.pdf
/usr/share/RevistaAtix/atix06.pdf
/usr/share/RevistaAtix/LEAME
/usr/share/RevistaAtix/TODO
/usr/share/RevistaAtix/COPYING
/usr/share/RevistaAtix/ChangeLog
/usr/share/applications/atix01.desktop
/usr/share/applications/atix02.desktop
/usr/share/applications/atix03.desktop
/usr/share/applications/atix04.desktop
/usr/share/applications/atix05.desktop
/usr/share/applications/atix06.desktop
/usr/share/pixmaps/atix01.xpm
/usr/share/pixmaps/atix02.xpm
/usr/share/pixmaps/atix03.xpm
/usr/share/pixmaps/atix04.xpm
```

```
/usr/share/pixmaps/atix05.xpm
/usr/share/pixmaps/atix06.xpm
```

%doc

Contiene el listado de archivos de documentación del paquete, éstos se instalarán en **/usr/share/doc/$NOMBRE-$VERSIÓN**

atix.spec (continuación)

```
%doc LEAME TODO COPYING ChangeLog
```

%changelog

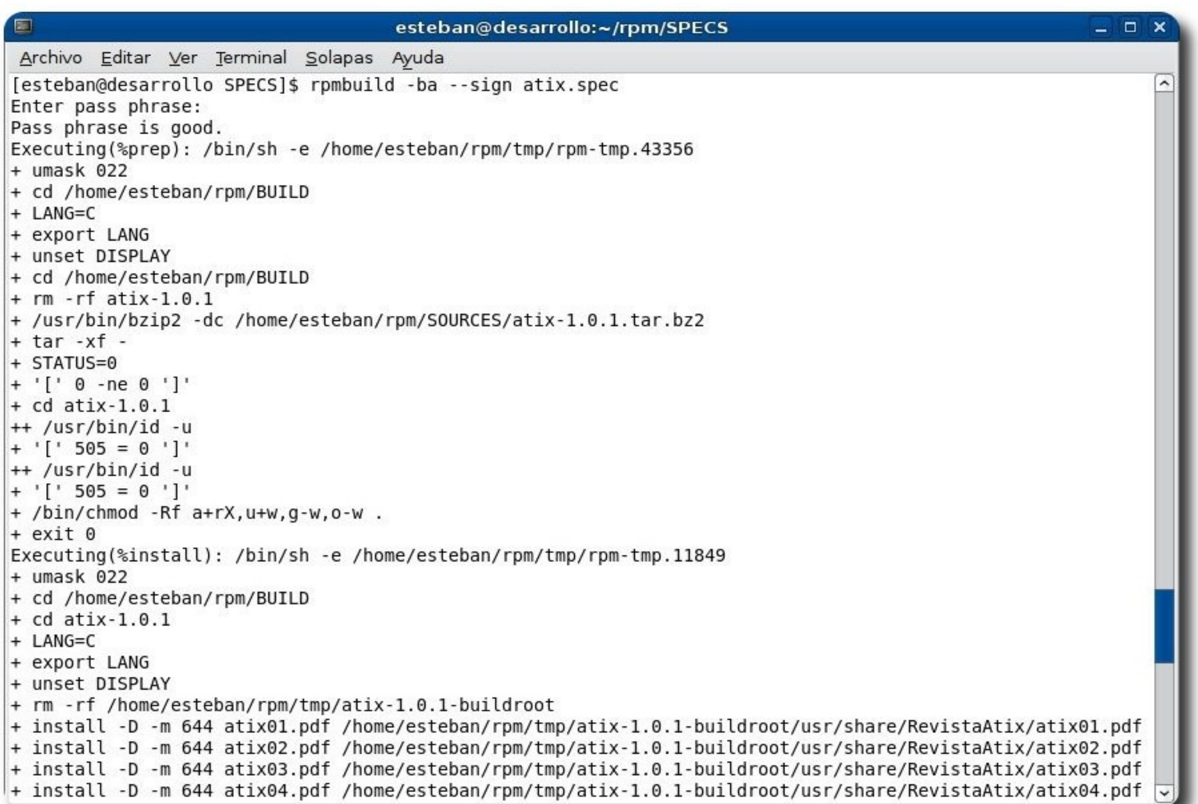
Contiene el detalle de los diversos cambios hechos al paquete.

atix.spec (continuación)

```
%changelog
* Wed Jan 7 2009 Esteban Saavedra Lopez <http://jesaavedra.opentelematics.org>
- Reempaquetado del paquete atix
- El paquete contiene los primeros 6 numeros de la revista Atix.
- Modificacion del archivo spec
* Tue Jan 6 2009 Esteban Saavedra Lopez <http://jesaavedra.opentelematics.org>
- Inicio del empaquetado del paquete atix
- El paquete solo contenia el primer numero de la revista Atix, por temas de prueba
- Creacion inicial del archivo spec
```

Creación del paquete

Una vez que se tiene la estructura de directorios adecuada, la firma digital, el fichero **.rpmmacros**, el archivo **.SPEC**, los archivos fuentes del paquete; podemos proceder a crear el paquete, como se muestra en la figura siguiente:



Detalles del paquete creado

```
[esteban@desarrollo ~]$ rpm -qi atix
Name        : atix                         Relocations: (not relocatable)
Version     : 1.0.1                             Vendor: Revista Atix <http://atix.opentelematics.org>
Release     : 1.revista                     Build Date: mié 07 ene 2009 11:24:57 BOT
Install Date: mié 07 ene 2009 11:25:32 BOT      Build Host: desarrollo.fps.gov.bo
Group       : Documentation                 Source RPM: atix-1.0.1-1.revista.src.rpm
Size        : 37927365                         License: GPL
Signature   : DSA/SHA1, mié 07 ene 2009 11:25:07 BOT, Key ID 487b87baa4bbfdbb
Packager    : Esteban Saavedra Lopez <estebansaavedra@yahoo.com>
URL         : http://atix.opentelematics.org/
Summary     : Numeros de la Revista Atix
Description :
atix es un paquete creado por Esteban Saavedra Lopez <jesaavedra@opentelematics.org>.
El objetivo de este paquete es instalar los distintos numeros publicados de la Revista de Software Lib
re Atix, y peridodicamente poseer actualizaciones que vayan incluyendo los distintos numeros segun se
vayan liberando.

Más información en: <http://atix.opentelematics.org>
[esteban@desarrollo ~]$
```

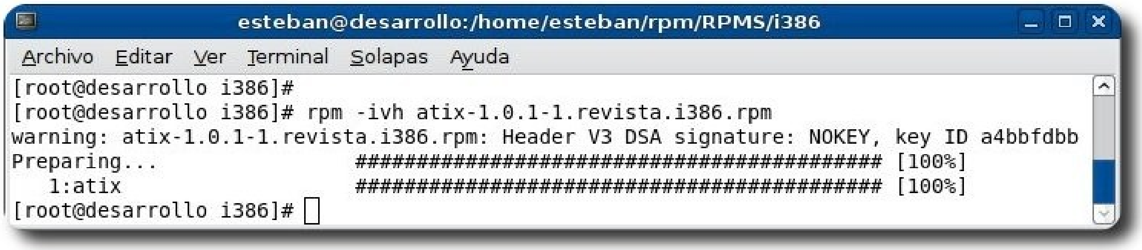
Información del paquete

```
[esteban@desarrollo ~]$ rpm -ql atix
/usr/share/RevistaAtix/COPYING
/usr/share/RevistaAtix/ChangeLog
/usr/share/RevistaAtix/LEAME
/usr/share/RevistaAtix/TODO
/usr/share/RevistaAtix/atix01.pdf
/usr/share/RevistaAtix/atix02.pdf
/usr/share/RevistaAtix/atix03.pdf
/usr/share/RevistaAtix/atix04.pdf
/usr/share/RevistaAtix/atix05.pdf
/usr/share/RevistaAtix/atix06.pdf
/usr/share/applications/atix01.desktop
/usr/share/applications/atix02.desktop
/usr/share/applications/atix03.desktop
/usr/share/applications/atix04.desktop
/usr/share/applications/atix05.desktop
/usr/share/applications/atix06.desktop
/usr/share/doc/atix-1.0.1
/usr/share/doc/atix-1.0.1/COPYING
/usr/share/doc/atix-1.0.1/ChangeLog
/usr/share/doc/atix-1.0.1/LEAME
/usr/share/doc/atix-1.0.1/TODO
/usr/share/man/man1/atix.1.gz
/usr/share/pixmaps/atix01.xpm
/usr/share/pixmaps/atix02.xpm
/usr/share/pixmaps/atix03.xpm
/usr/share/pixmaps/atix04.xpm
/usr/share/pixmaps/atix05.xpm
/usr/share/pixmaps/atix06.xpm
[esteban@desarrollo ~]$
```

Listado de archivos del paquete

Instalación del paquete

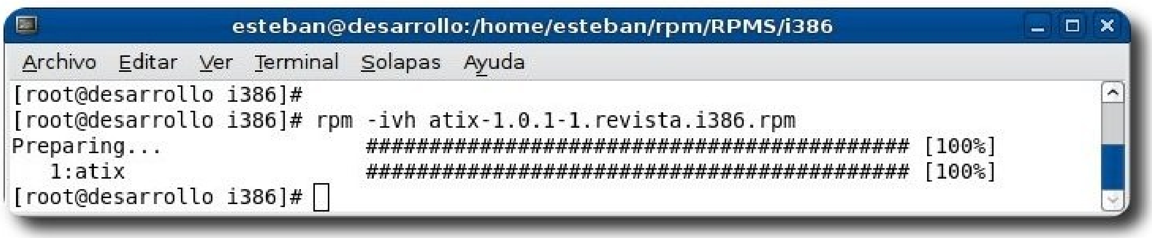
Cuando procedemos a instalar un paquete que no tenga una firma digital el sistema operativo nos advierte de éste echo como muestra la figura.



Por eso es conveniente instalar paquetes que dispongan de una firma digital que garantice su autenticidad, en nuestro caso primero procedemos a importar la llave como muestra la figura



y luego procedemos a instalar el paquete, donde observaremos que el sistema operativo ya no muestra la advertencia anteriormente vista.



Demostración del paquete instalado

Uno de los objetivos que teníamos, era que después de instalado el paquete creado, éste permita el acceso desde el menú de aplicaciones a todos y cada uno de los números de la Revista Atix; esto se logra con ayuda de los archivos atix01.desktop, ..atix06.desktop. El contenido de uno de estos archivos es el mostrado a continuación:

atix01.desktop

```
[Desktop Entry]
Name=Revista Atix No 1
Comment=Revista de Software Libre Atix No 1
Categories=Education;
Encoding=UTF-8
Exec=evince /usr/share/RevistaAtix/atix01.pdf
Icon=atix01
StartupNotify=true
Terminal=false
Type=Application
```

La siguiente figura muestra como se ve el menú aplicaciones, incluyendo las opciones referidas a cada número de la revista.



Trabajo futuro

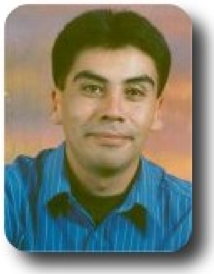
Un trabajo que se debería realizar es la creación de un repositorio público en Internet, que permita a cualquier usuario poder instalar y actualizar el paquete atix-1.0.1-1.revista.noarch.rpm, haciendo uso del gestor de paquetes yum y referenciado por un archivo atix.repo (para el caso de RHEL, CentOs, Fedora)

Referencias

[1] http://www.rpm.org/

[2] http://es.wikipedia.com

Autor



Esteban Saavedra López
Líder de la Comunidad ATIX (Oruro – Bolivia)
Activista de Software Libre en Bolivia
jesaavedra@opentelematics.org
http://jesaavedra.opentelematics.org

Gestión de Certificados Digitales con OpenSSL (1ra parte)

Hoy por hoy el uso de certificados digitales se ha hecho tan frecuente e importante, sobre todo la momento de garantizar la privacidad y seguridad tanto en el intercambio de documentos como en establecer comunicaciones seguras en los distintos servicios que hacen uso del Internet.

Introducción

Hoy en día es tan frecuente e importante hacer uso de servicios basados en Internet (correo electrónico, servicios web, transacciones en línea y muchos más), razón por la cual los proveedores de estos servicios se ven en la necesidad de proporcionar una manera de garantizar la privacidad y seguridad de éste tipo de comunicaciones.

Una manera de garantizar la comunicación entre dos extremos es cifrar la comunicación (establecer un canal seguro), haciendo uso de criptografía simétrica y asimétrica, considerando también que muchos servicios utilizan criptografía de llave pública usando certificados SSL x509, todo con el fin de asegurar la identidad y confidencialidad de los usuarios.

Cabe destacar que existen instituciones internacionales, cuya principal función es la de brindar servicios comerciales para expedir certificados SSL, tal el caso de Thawte y VeriSign.

En la actualidad no necesariamente debemos acudir a estas instituciones para poseer un certificado digital que nos sirva para firmar documentos o cifrar nuestras comunicaciones, ya que podemos hacer uso de alternativas libres que tienen las mismas funciones pero no hay que pagar por expedir los certificados necesarios para nuestros propósitos, tal es el caso de **OpenSSL**.

OpenSSL

OpenSSL es una implementación libre, de código abierto, de los protocolos SSL (Secure Sockets Layer o Nivel de Zócalo Seguro) y TLS (*Transport Layer Security*, o *Seguridad para Nivel de Transporte*).

Características

- ✔ Proyecto de Código Abierto
- ✔ Licencia Libre
- ✔ Cifrado Fuerte (3DES, Blowfish)
- ✔ Reenvío por X11 (cifra el tráfico de X Window System)
- ✔ Reenvío por Puertos (canales cifrados por protocolos de legado)
- ✔ OpenSSL, tiene soporte para el protocolo SSH 1.3 , SSH 1.5 , SSH 2.0
- ✔ Reenvío por Agente
- ✔ Soporte para cliente y servidor de SFTP en los protocolos SSH1 y SSH2.
- ✔ Pases de Ticket de Kerberos y AFS en la maquina remota
- ✔ Autenticación Fuerte (Clave Pública,
- ✔ Compresión de datos

Conceptos útiles

A continuación presentamos algunos conceptos que serán útiles al momento de comprender la gestión de certificados digitales.

- **Certificado digital**: Un Certificado Digital es un documento digital mediante el cual un tercero confiable (una autoridad de certificación) garantiza la vinculación entre la identidad de un sujeto o entidad y su clave pública.
- **Autoridad certificadora** : Una Autoridad de certificación, certificadora o certificante (AC o CA por sus siglas en inglés Certification Authority) es una entidad de confianza, responsable de emitir y revocar los certificados digitales o certificados, utilizados en la firma electrónica, para lo cual se emplea la criptografía de clave pública. Jurídicamente es un caso particular de Prestador de Servicios de Certificación.
- **Firma digital**: La firma digital o firma electrónica es, en la transmisión de mensajes telemáticos y en la gestión de documentos electrónicos, un método criptográfico que asocia la identidad de una persona o de un equipo informático al mensaje o documento. En función del tipo de firma, puede, además, asegurar la integridad del documento o mensaje.

  Un certificado digital además de estar firmado digitalmente por ésta, debe contener por lo menos lo siguiente:

  - Nombre, dirección y domicilio del suscriptor.
  - Identificación del suscriptor nombrado en el certificado.
  - El nombre, la dirección y el lugar donde realiza actividades la entidad de certificación.
  - La clave pública del usuario.
  - La metodología para verificar la firma digital del suscriptor impuesta en el mensaje de datos.
  - El número de serie del certificado.
  - Fecha de emisión y expiración del certificado.
- **SSL** : Secure Sockets Layer, es seguridad de la capa de transporte que proporcionan comunicación segura a internet ssl utiliza autenticación y privacidad.
- **Certificados x.509** : x.509 es un estándar para una Infraestructura de Llave Publica (PKI), el cual especifíca el formato para certificados de claves públicas, y un algoritmo de validación de la ruta del certificado. La versión actual de x509 es la versión 3 y es la que se usa ampliamente.

  Los datos de un sujeto que se incluyen en un certificado x.509 son:

  - CN: nombre común o nombre largo
  - E-Mail: Dirección de correo electrónico
  - O: Nombre de su organización
  - OU: Departamento
  - L: Localidad
  - SP: Provincia o estado.
  - C: País

Descripción de archivos generados

- **cakey.pem**: clave privada de la autoridad certificadora.
- **cacert.pem**: certificado de la autoridad certificadora.
- **cliente.key**: clave del servidor para el que se solicita el certificado.
- **cliente.csr**: solicitud de certificado del servidor.
- **cliente.crt**: certificado del servidor, firmado por la CA.

La extensión "pem" es de *Privacy Enhanced Message*

Aplicación

Las aplicaciones de los certificados digitales, son variadas al momento de utilizar certificados digitales, entre las que destacan:

- ✔ Brindar seguridad en la capa de transporte par servicios variados como: web(https), correo electrónico (pops,imaps), redes privadas virtuales, etc
- ✔ Firma de documentos

Que precisamos

Básicamente se debe contar con la instalación de **OpenSSL**, y alguna paquetería adicional dependiendo del servicio con el que se interactúe (mod_ssl para el caso de apache por ejemplo).

La instalación de **OpenSSL** y de sus dependencia se la puede realizar por medio del gestor de paquetes de la distribución sobre la cual se esta trabajando:

- ✔ **yum**, para el caso de RHEL, CentOS y Fedora
- ✔ **apt-get**, para el caso de Debian y Ubuntu

Estructura de un certificado

La estructura de un certificado digital X.509 v3 es la siguiente:

- ✔ Certificado
  - ✔ Versión
  - ✔ Número de serie
  - ✔ ID del algoritmo
  - ✔ Emisor
  - ✔ Validez
    - ✔ No antes de
    - ✔ No después de
  - ✔ Tema
  - ✔ Tema información de clave pública
    - ✔ Algoritmo de clave pública
    - ✔ Tema clave pública
  - ✔ Identificador único de emisor (opcional)
  - ✔ Identificador único de tema (opcional)
  - ✔ Extensiones (opcional)
- ✔ Algoritmo de certificado de firma
- ✔ Certificado de firma

Los identificadores únicos de emisor y tema fueron introducidos en la versión 2, y las extensiones en la versión 3.

Configuraciones base

En caso de ser necesario el contar con una configuración especifica que permita crear o firmar los certificados bajo ciertas características, es preciso contar con un archivo de configuración, generalmente llamado `openSSL.cnf`.

Gestión de certificados

Dentro la gestión de certificados digitales, se deben considerar los siguientes pasos:

- ✔ Crear la estructura del servidor de certificación y archivo de configuración
- ✔ Iniciar la CA
- ✔ Generar la solicitud de certificados
- ✔ Firmar los certificados
- ✔ Algunas operaciones comunes con certificados
  - ✔ Revocar los certificados
  - ✔ Obtener información de certificados
  - ✔ Conversión de formatos de los certificados.

Crear la estructura del servidor de certificación

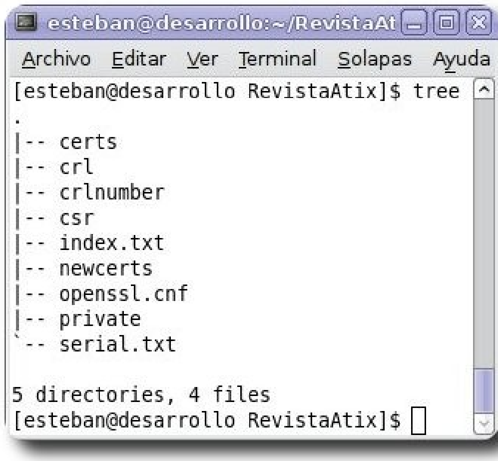
Para mantener cierto orden en la gestión de certificados es recomendable mantener una estructura de directorios que permita mantener clasificados los distintos elementos inmersos en el proceso de gestión de certificación. la estructura recomendada es la siguiente:

```
nombre_institucion/
nombre_institucion/certs
nombre_institucion/private
nombre_institucion/newcerts
nombre_institucion/csr
nombre_institucion/crl
nombre_institucion/OpenSSL.cnf
nombre_institucion/index.txt
nombre_institucion/serial.txt
nombre_institucion/crlnumber
```

donde:

- ✔ **certs:** directorio para contener certificados
- ✔ **csr:** directorio para contener los archivos de solicitud de certificados
- ✔ **crl:** directorio para contener certificados revocados
- ✔ **index.txt:** fichero con el índice de certificados firmados
- ✔ **newcerts:** directorio para contener los nuevos certificados emitidos
- ✔ **private:** directorio que contiene el fichero cakey.pem, la llave privada
- ✔ **serial.txt:** fichero que contiene el número de serie de certificados firmados.
- ✔ **crlnumber:** fichero que contiene el número de serie de certificados revocados.

Para nuestro caso la estructura de directorios es la siguiente:



Archivo de configuración

Toda vez que deseamos hacer uso de **OpenSSL** para la gestión de certificados digitales, existe la posibilidad de brindar algunas configuraciones por defecto, éstas se deben incluir en un archivo, el cual sea referenciado al momento de gestionar los certificados digitales.

Este archivo (**`openssl.cnf`**) generalmente se encuentra en **`/etc/pki/tls/`** en el caso de CentOS y en **`/etc/ssl`** en el caso de Ubuntu; para trabajar con este archivo es recomendable obtener una copia del mismo y localizarlo en la estructura de directorios creada anteriormente.

Aquí les presentamos las secciones que generalmente son necesarias adecuar:

```
#################################################################### [ ca ]
default_ca      = CA_default            # The default ca section
####################################################################
[ CA_default ]
dir             = /home/esteban/RevistaAtix     # Where everything is kept
certs           = $dir/certs            # Where the issued certs are kept
crl_dir         = $dir/crl              # Where the issued crl are kept
database        = $dir/index.txt        # database index file.
new_certs_dir   = $dir/newcerts         # default place for new certs.
```

```
certificate     = $dir/cacert.pem       # The CA certificate
serial          = $dir/serial.txt               # The current serial number
crlnumber       = $dir/crlnumber        # the current crl number
                                        # must be commented out to leave a V1 CRL
crl             = $dir/crl.pem          # The current CRL
private_key     = $dir/private/cakey.pem        # The private key
RANDFILE        = $dir/private/.rand    # private random number file

x509_extensions = usr_cert              # The extentions to add to the cert

# Comment out the following two lines for the "traditional"
# (and highly broken) format.
name_opt        = ca_default            # Subject Name options
cert_opt        = ca_default            # Certificate field options
default_days    = 365                   # how long to certify for
default_crl_days= 30                    # how long before next CRL
default_md      = sha1                  # which md to use.
preserve        = no                    # keep passed DN ordering
# For the CA policy
[ policy_match ]
countryName             = match
stateOrProvinceName     = match
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

# For the 'anything' policy
# At this point in time, you must list all acceptable 'object'
# types.
[ policy_anything ]
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

####################################################################
[ req ]
default_bits            = 1024
default_md              = sha1
default_keyfile         = privkey.pem
distinguished_name      = req_distinguished_name
attributes              = req_attributes
# req_extensions = v3_req # The extensions to add to a certificate request

[ req_distinguished_name ]
countryName                     = Codigo del pais (dos letras)
countryName_default             = BO
countryName_min                 = 2
countryName_max                 = 2

stateOrProvinceName             = Ciudad o departamento
stateOrProvinceName_default     = Oruro

localityName                    = Nombre Localidad
localityName_default            = Oruro

0.organizationName              = Nombre Organizacion
0.organizationName_default      = Revista Atix
```

```
organizationalUnitName           = Nombre Unidad Organizacional
organizationalUnitName_default   = Direccion y Cordinacion General

commonName                       = Nombre Comun (eg, nombre del servidor)
commonName_default               = atix.opentelematics.org
commonName_max                   = 64

emailAddress                     = Direccion de Email
emailAddress_default             = security@atix.opentelematics.org
emailAddress_max                 = 64
[ req_attributes ]
challengePassword                = Una clave de seguridad
challengePassword_min            = 4
challengePassword_max            = 20
```

Dentro de esta configuración se hace referencia a 2 archivos que deben ser creados previamente, de la forma siguiente:

echo "01" > serial .txt

touch index.txt

echo "01" > crlnumber

Iniciar la CA

Antes de actuar en una entidad certificadora (CA), primero debemos:

- ✔ Crear la llave con la que firmaremos los certificados
- ✔ Crear un certificado autorfirmado

```
esteban@desarrollo:~/RevistaAtix
Archivo  Editar  Ver  Terminal  Solapas  Ayuda
[esteban@desarrollo RevistaAtix]$ openssl req -new -x509 -days 3650 -keyout private/cakey.pem -out
cacert.pem -config openssl.cnf
Generating a 1024 bit RSA private key
...........++++++
.....++++++
writing new private key to 'private/cakey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Codigo del país (dos letras) [BO]:
Ciudad o departamento [Oruro]:
Nombre Localidad [Oruro]:
Nombre Organizacion [Revista Atix]:
Nombre Unidad Organizacional [Dirección y Cordinación General]:
Nombre Comun (eg, nombre del servidor) [atix.opentelematics.org]:
Dirección de Email [security@atix.opentelematics.org]:
[esteban@desarrollo RevistaAtix]$
[esteban@desarrollo RevistaAtix]$
```

Esto creará dos archivos,

- ✔ **cacert.pem**: el certificado raíz, el cual puede ser publicado a los que vayan a usar nuestra infraestructura de llave pública.
- ✔ **cakey.pem**: el archivo de la llave privada, el cual se usará para firmar las Certificados de Requerimiento de Certificados (CSR), este archivo se debe tener en un lugar seguro.

Apariencia de los certificados

Esta es la apariencia que tienen los archivos generados anteriormente:

cacert.pem


```
-----BEGIN CERTIFICATE-----
MIIEQDCCA6mgAwIBAgIJAIEX94lnv5uhMA0GCSqGSIb3DQEBBQUAMIHHMQswCQYD
VQQGEwJCTzEOMAwGA1UECBMFT3J1cm8xDjAMBgNVBAcTBU9ydXJvMRUwEwYDVQQK
EwxSZXZpc3RhIEF0aXgxLjAsBgNVBAsMJURpcmVjY2nDg8KzbiB5IENvcmRpbmFj
acODwrNuIEdlbmVyYWwxIDAeBgNVBAMTF2F0aXgub3BlbnRlbGVtYXRpY3Mub3Jn
MS8wLQYJKoZIhvcNAQkBFiBzZWN1cml0eUBhdGl4Lm9wZW50ZWxlbWF0aWNzLm9y
ZzAeFw0wODEyMjkxODI5NDRaFw0xODEyMjcxODI5NDRaMIHHMQswCQYDVQQGEwJC
TzEOMAwGA1UECBMFT3J1cm8xDjAMBgNVBAcTBU9ydXJvMRUwEwYDVQQKEwxSZXZp
c3RhIEF0aXgxLjAsBgNVBAsMJURpcmVjY2nDg8KzbiB5IENvcmRpbmFjacODwrNu
IEdlbmVyYWwxIDAeBgNVBAMTF2F0aXgub3BlbnRlbGVtYXRpY3Mub3JnMS8wLQYJ
KoZIhvcNAQkBFiBzZWN1cml0eUBhdGl4Lm9wZW50ZWxlbWF0aWNzLm9yZzCBnzAN
BgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA5iZMRd8wirUuKJi1pjy39YoPWuSgOtaY
fzeujQ0qt63BZkL3ao7DrkwGne8B1X0Jt6HEJjqoI7OxVGvHM66828FUI1APkJmu
5Y4NS1L8G8xx0BOX0J+vXvZXjAibcCZF6NWwFp4G1UVcVjZnR4f6xG51RyMqaD8s
gQxzOpdIm8sCAwEAAaOCATAwggEsMB0GA1UdDgQWBBT70SCZE0p4dSnq+g6VhkWG
lKHFCTCB/AYDVR0jBIH0MIHxgBT70SCZE0p4dSnq+g6VhkWGlKHFCaGBzaSByjCB
xzELMAkGA1UEBhMCQk8xDjAMBgNVBAgTBU9ydXJvMQ4wDAYDVQQHEwVPcnVybzEV
MBMGA1UEChMMUmV2aXN0YSBBdGl4MS4wLAYDVQQLDCVEaXJlY2Npw4PCs24geSBD
b3JkaW5hY2nDg8KzbiBHZW5lcmFsMSAwHgYDVQQDExdhdGl4Lm9wZW50ZWxlbWF0
aWNzLm9yZzEvMC0GCSqGSIb3DQEJARYgc2VjdXJpdHlAYXRpeC5vcGVudGVsZW1h
dGljcy5vcmeCCQCBF/eJZ7+boTAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBBQUA
A4GBAB5pLGkJ+X1yRFMzn97CBs87Ox3f9IZJuYvrGJ3pyILJ+nlHALNBSCcglViG
Ul0tREu5X6hlfNDjw2NmUJer1k8p6MAyreZ6qquFTNhPloozNiCdqMSMhodRQrt4
tLyK3Sz75teRkcOS/zXbzxoY/igtFz/qRSgCxftwzvijk3zU
-----END CERTIFICATE-----
```

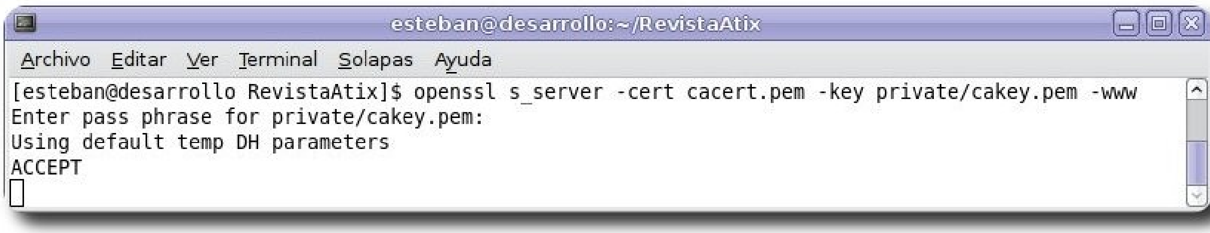

private/cakey.pem


```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,056E2D7558454992

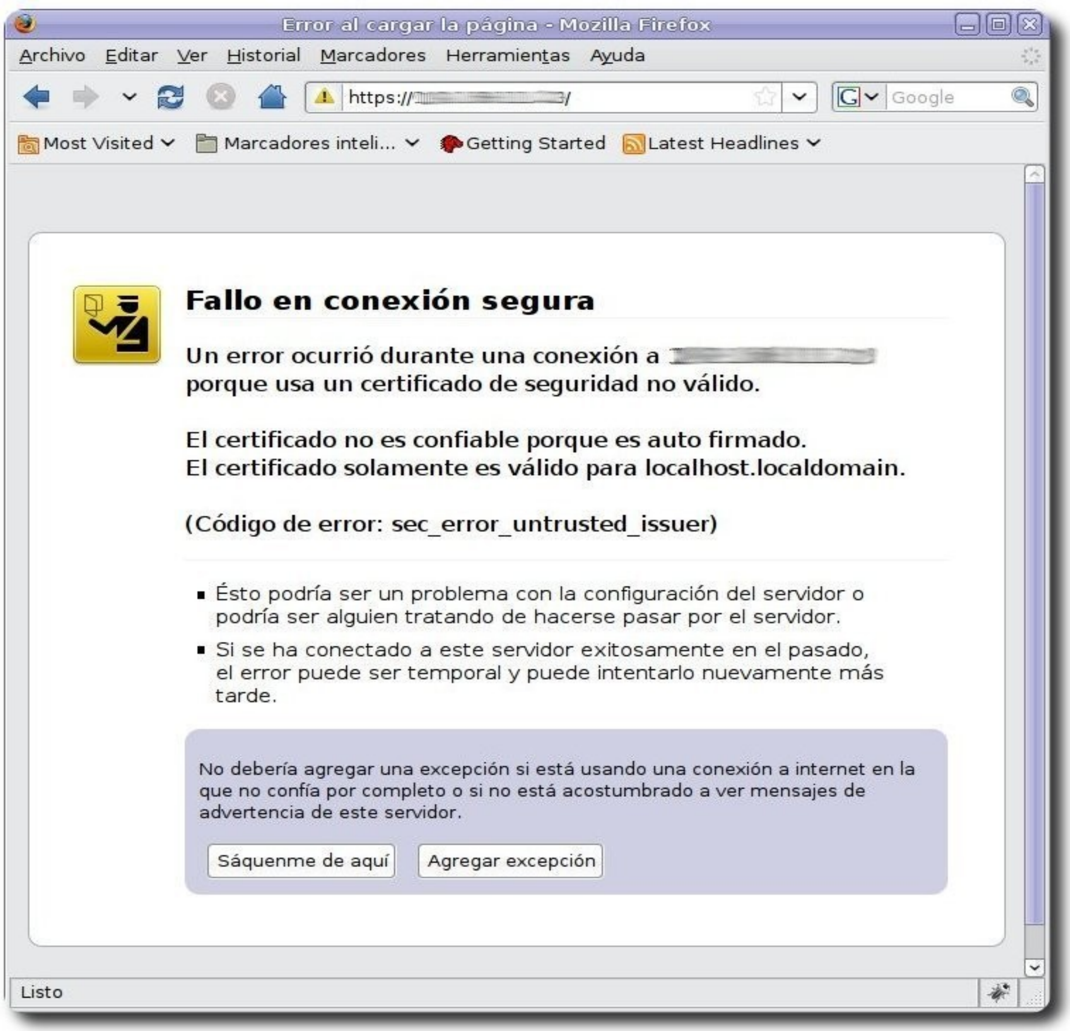
Oed3C62YrL7v2dT9U6yikJHJootNvsFQEmFSAT1kOCOYGBzYp806sCbS/9nHNhk6
c6l0Vv0c6Quok4F+jtrLbpLpCQxVxd1HUeyFMs3tcuTBU68c3WP8C0en81hAicjv
ynScGhE30W1x5jqIL98T06Rk8EG9zleeVivE/d98jVPBp9BCh1eCKMNobvLvQKxD
/5lzQJe427dib2Ty4mICRDIChvP5wK36O0BkB3pVeY0oWXrBLK+OJ7AWG6BAkwg6
Vts1BYm3bf3MhFc8l38v9kj3WOcfshyAJfpiKiMN2Y7gnTHhtN36LMq34wBaWGcS
qZ7Sa3sfuRSLIQB+pApK+kUZ9Mqg0gxCQvPZyqNsq0gzWidL4OZpx8Kp086XWZZf
Tak7eE4HmanzMxXb/sgXZF7OgziCMSF4HmA2/SXlRsfWo9aiVT181c8LQmgXSa7F
PjABhMGUBmUAGSlHv5a1l2SQgijqL01gsSVhc7aTYyPEyZFd1/cfjpCTpXPGa/Z0
0p+8wF/LIH8SK+KpWSXRXqWqDJtx3kqv24YKox5szWaDr01x1y1Ib5F0UBorDLDJ
uwl+6iQpnLEbp6u3bTr82BHckPbpPMVldK+Xo/BOHOWCdnNbkpGs0ODQbHEIWyPA
cuyhe+LBMrIinMsnVz5rcNh/KtJ5MVsPsx3jeKQfXjwdqaAdy9/u+t6h6SjlmUMb
LNTRnJBjBplbQ6VTtDn/9vOJx5Bpu0UG/DEnl1eLByFPABY+R3sh3ZDAiDg0ZCMl
Fjuq4oDrhqcKbsM5mehFyyzXVm9QQLHjfGxa4uvXQoByttwmwQBKug==
-----END RSA PRIVATE KEY-----
```

Poner a prueba los certificados

Aunque el certificado raíz no será usado para algún servicio, podemos comprobar su funcionamiento, de la siguiente forma:

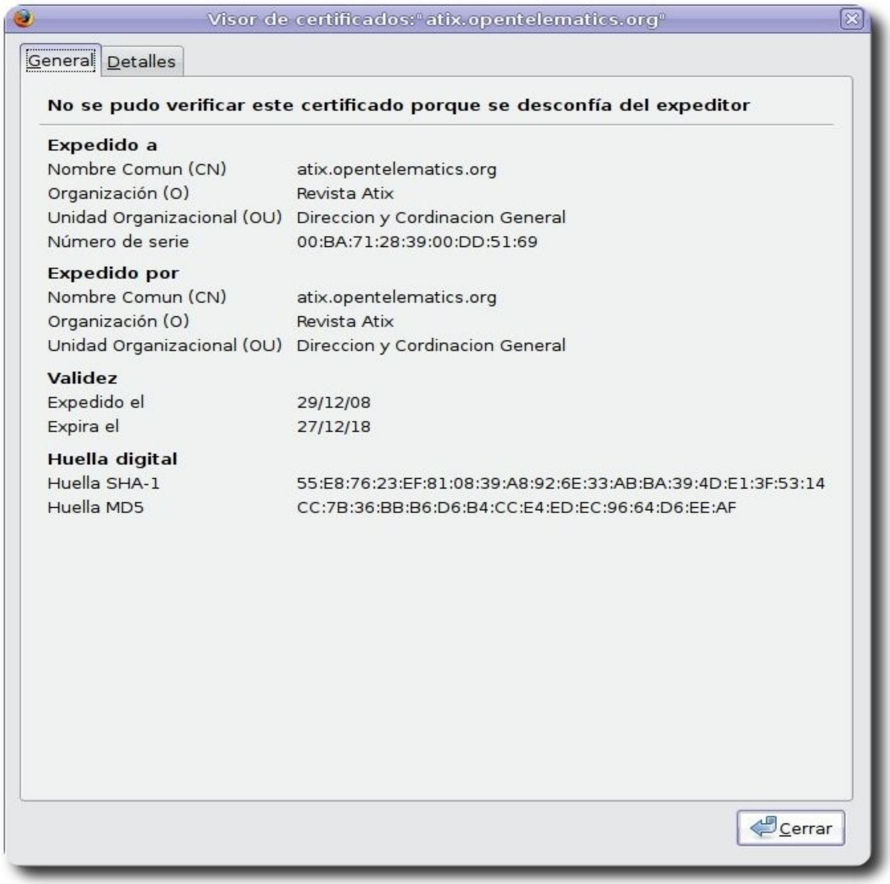


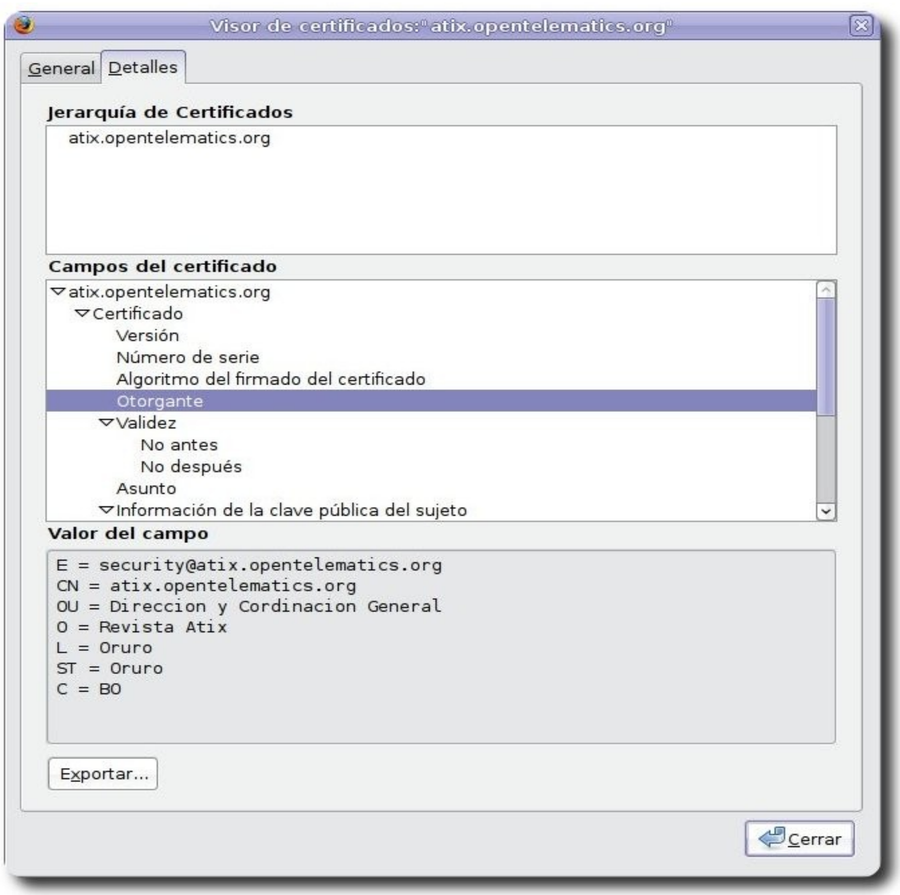
Para ver el resultado, podemos acceder al sitio mediante un browser.



Siguiendo la opción de "*Agregar la excepción*", podemos recuperar, ver e incluir el certificado dentro del browser y de ésta forma poder establecer una conexión segura vía http.

La información y los detalles del certificado pueden ser vistos antes de ser agregados, como muestran las figuras siguientes:


Visor de certificados:"atix.opentelematics.org"
General
Detalles
No se pudo verificar este certificado porque se desconfía del expeditor
Expedido a
Nombre Comun (CN) atix.opentelematics.org
Organización (O) Revista Atix
Unidad Organizacional (OU) Direccion y Cordinacion General
Número de serie 00:BA:71:28:39:00:DD:51:69
Expedido por
Nombre Comun (CN) atix.opentelematics.org
Organización (O) Revista Atix
Unidad Organizacional (OU) Direccion y Cordinacion General
Validez
Expedido el 29/12/08
Expira el 27/12/18
Huella digital
Huella SHA-1 55:E8:76:23:EF:81:08:39:A8:92:6E:33:AB:BA:39:4D:E1:3F:53:14
Huella MD5 CC:7B:36:BB:B6:D6:B4:CC:E4:ED:EC:96:64:D6:EE:AF
Cerrar


Visor de certificados:"atix.opentelematics.org"
General
Detalles
Jerarquía de Certificados
atix.opentelematics.org
Campos del certificado
atix.opentelematics.org
Certificado
Versión
Número de serie
Algoritmo del firmado del certificado
Otorgante
Validez
No antes
No después
Asunto
Información de la clave pública del sujeto
Valor del campo
E = security@atix.opentelematics.org
CN = atix.opentelematics.org
OU = Direccion y Cordinacion General
O = Revista Atix
L = Oruro
ST = Oruro
C = BO
Exportar...
Cerrar

Creación de un CSR usando OpenSSL

Los archivos CSR (Certificados de Requerimiento de Certificados), son solicitudes de certificados generados por las personas o instituciones, archivos que al ser firmados por una CA serán certificados, que podrán ser utilizados en algún servicio de Internet.

```
joseph@desarrollo:~/CentroDesarrollo
Archivo  Editar  Ver  Terminal  Solapas  Ayuda
[joseph@desarrollo CentroDesarrollo]$ openssl req -nodes -new -keyout private/desarrollo.dominio.co
m.key -out csr/desarrollo.dominio.com.csr -config openssl.cnf
Generating a 1024 bit RSA private key
..............++++++
..++++++
writing new private key to 'private/desarrollo.dominio.com.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Codigo del país (dos letras) [BO]:
Ciudad o departamento [La Paz]:
Nombre Localidad [La Paz]:
Nombre Organizacion [Centro de Desarrollo]:
Nombre Unidad Organizacional [Jefatura de desarrollo]:
Nombre Comun (eg, nombre del servidor) [desarrollo.dominio.com]:
Direccion de Email [desarrollo@dominio.com]:

Please enter the following 'extra' attributes
to be sent with your certificate request
Una clave de seguridad []:desarrollo
[joseph@desarrollo CentroDesarrollo]$
```

La instrucción anterior creará un par de archivos, tal como se muestra en la figura siguiente:

```
joseph@desarrollo:~/CentroDesarrol
Archivo  Editar  Ver  Terminal  Solapas  Ayuda
[joseph@desarrollo CentroDesarrollo]$ tree
.
|-- certs
|-- crl
|-- csr
|   `-- desarrollo.dominio.com.csr
|-- index.txt
|-- newcerts
|-- openssl.cnf
|-- private
|   `-- desarrollo.dominio.com.key
`-- serial.txt

5 directories, 5 files
[joseph@desarrollo CentroDesarrollo]$
```

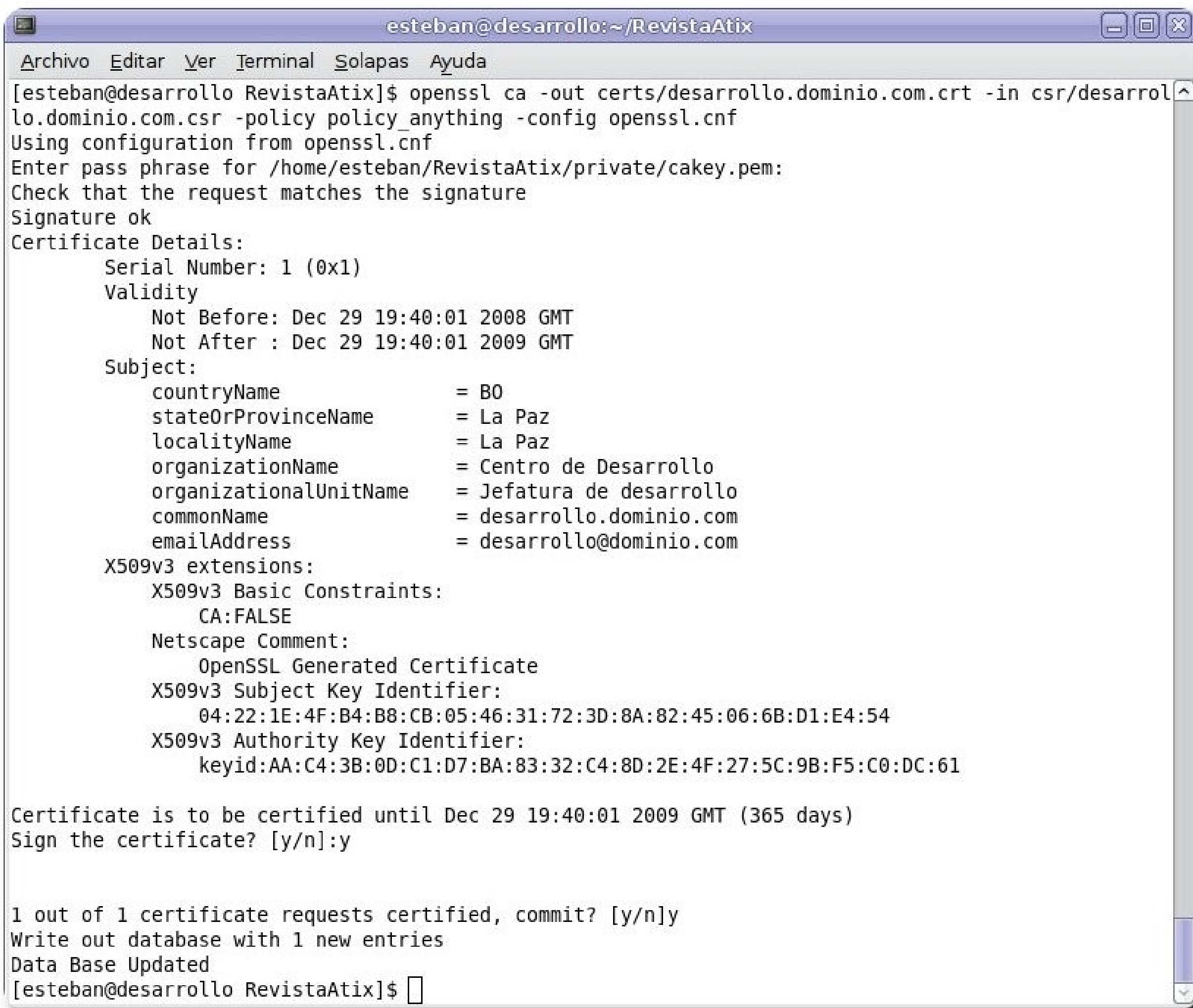
El archivo de solicitud (.csr) tiene el siguiente contenido:

desarrollo.dominio.com.csr


```
-----BEGIN CERTIFICATE REQUEST-----
MIICEzCCAXwCAQAwgbcxCzAJBgNVBAYTAkJPMQ8wDQYDVQQIEwZMYSBQYXoxDzAN
BgNVBAcTBkxhIFBhejEdMBsGA1UEChMUQ2VudHJvIGRlIERlc2Fycm9sbG8xHzAd
BgNVBAsTFkplZmF0dXJhIGRlIGRlc2Fycm9sbG8xHzAdBgNVBAMTFmRlc2Fycm9s
bG8uZG9taW5pby5jb20xJTAjBgkqhkiG9w0BCQEWFmRlc2Fycm9sbG9AZG9taW5p
by5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBANs1IPRyVYn5UkzdDQ5G
ik0yFBL3B1JI00gqluEFLPJrFnbBtYHz7HsVMAFYWo3bLDvX4vekzZH8G+70n2Sf
neZy0jSnn5AYqDDzpH7c2H1juf0Ze3BalZyTwb7tpxRuj+BWrC5JNt/Yv6cPkpm/
Fv18C7yg06fGvnV+2EJVgomvAgMBAAGgGzAZBgkqhkiG9w0BCQcxDBMKZGVzYXJy
b2xsbzANBgkqhkiG9w0BAQUFAAOBgQCeq+7oc4ZBeMF4LVpPQqiI6a6XovJiC0Gv
DnPjoN3BHUi8JpXHYWNJ3DPI+yKaquNtYzcrYmHkE/beXjUXga9RTG6XVcoS0wDK
nLyg0b8Cgz6KWFg+dIxYIvyAAAq+yd7CCdmpw2+2gcMMRd27X6TcWP3olk6mS/l/
p+XgscD4JA==
-----END CERTIFICATE REQUEST-----
```

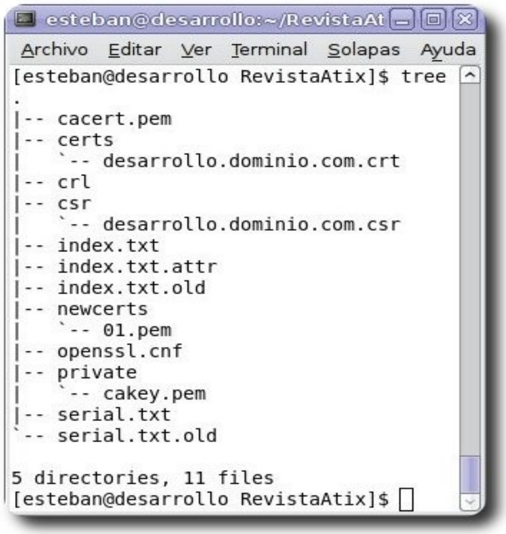

Creación y/o firma de certificados

Una vez que la CA recibe el CSR del cliente, se puede proceder a firmarlo y de esta forma convertirlo en un certificado , tal como se muestra en la figura siguiente:



Una vez que se procedió a firmar el archivo CSR, se generará un certificado SSL correspondiente, que deberá ser devuelto al cliente.

Por defecto **OpenSSL** guarda una copia de el certificado para un uso futuro, como la revocación del certificado. Los certificados son almacenados bajo el directorio newcerts, pero con el nombre de archivo basados en el número de serie, y no en el nombre del host de la petición.



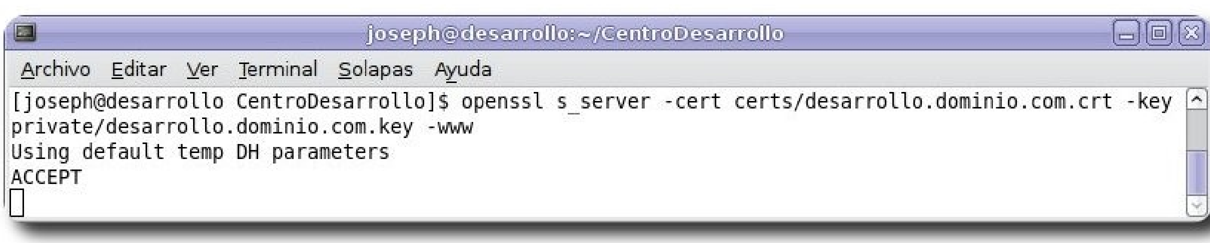
La apariencia del archivo (crt) correspondiente al certificado SSL presenta la siguiente apariencia:

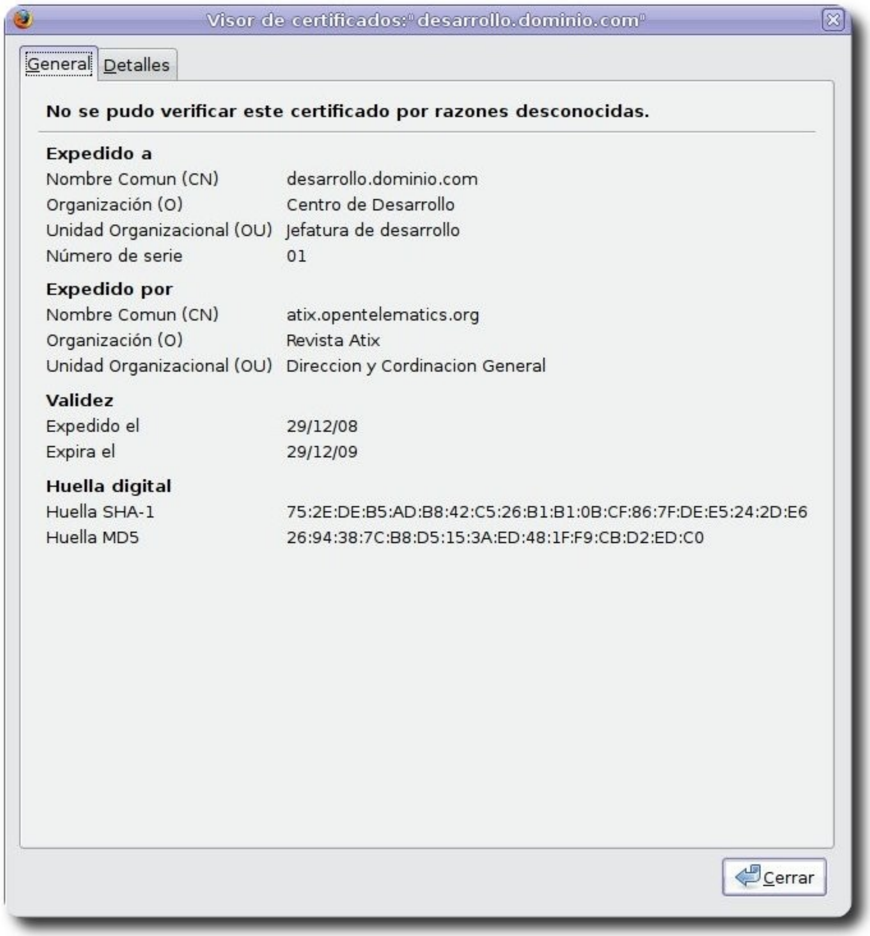
desarrollo.dominio.com.crt

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=BO, ST=Oruro, L=Oruro, O=Revista Atix, OU=Direccion y Cordinacion
General, CN=atix.opentelematics.org/emailAddress=security@atix.opentelematics.org
        Validity
            Not Before: Dec 29 19:40:01 2008 GMT
            Not After : Dec 29 19:40:01 2009 GMT
        Subject: C=BO, ST=La Paz, L=La Paz, O=Centro de Desarrollo, OU=Jefatura de
desarrollo, CN=desarrollo.dominio.com/emailAddress=desarrollo@dominio.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:db:35:20:f4:72:55:89:f9:52:4c:dd:0d:0e:46:
                    8a:4d:32:14:12:f7:07:52:48:d3:48:2a:96:e1:05:
                    2c:f2:6b:16:76:c1:b5:81:f3:ec:7b:15:30:01:58:
                    5a:8d:db:2c:3b:d7:e2:f7:a4:cd:91:fc:1b:ee:f4:
                    9f:64:9f:9d:e6:72:d2:34:a7:9f:90:18:a8:30:f3:
                    a4:7e:dc:d8:7d:63:b9:fd:19:7b:70:5a:95:9c:93:
                    c1:be:ed:a7:14:6e:8f:e0:56:ac:2e:49:36:df:d8:
                    bf:a7:0f:92:99:bf:16:fd:7c:0b:bc:a0:d3:a7:c6:
                    be:75:7e:d8:42:55:82:89:af
                Exponent: 65537 (0x10001)
```

```
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                04:22:1E:4F:B4:B8:CB:05:46:31:72:3D:8A:82:45:06:6B:D1:E4:54
            X509v3 Authority Key Identifier:
                keyid:AA:C4:3B:0D:C1:D7:BA:83:32:C4:8D:2E:4F:27:5C:9B:F5:C0:DC:61
   Signature Algorithm: sha1WithRSAEncryption
        61:2e:64:d5:36:2d:7a:50:ae:de:25:cd:ef:60:02:f4:52:2c:
        ae:ac:65:b0:d6:a9:b3:b2:30:a4:36:07:ad:b3:2f:32:ce:e9:
        ea:84:5e:fd:0d:b5:97:7c:d8:7b:99:a2:ce:ce:64:a9:34:01:
        5e:91:f3:7d:50:a7:7e:71:91:cb:5d:67:60:46:ac:23:4d:81:
        ac:e2:fe:3d:32:56:cc:c3:ae:e0:7a:a7:29:57:bb:a1:cb:5e:
        6b:71:43:2a:78:e8:fe:84:13:de:72:0f:2f:ff:78:6d:ff:68:
        a9:ff:86:da:30:23:6f:f1:f4:0c:e4:cf:e6:54:02:b7:ad:9f:
        3e:11
-----BEGIN CERTIFICATE-----
MIIDazCCAtSgAwIBAgIBATANBgkqhkiG9w0BAQUFADCBwTELMAkGA1UEBhMCQk8x
DjAMBgNVBAgTBU9ydXJvMQ4wDAYDVQQHEwVPcnVybzEVMBMGA1UEChMMUmV2aXN0
YSBBdGl4MSgwJgYDVQQLEx9EaXJlY2Npb24geSBDb3JkaW5hY2lvbiBHZW5lcmFs
MSAwHgYDVQQDExdhdGl4Lm9wZW50ZWxlbWF0aWNzLm9yZzEvMC0GCSqGSIb3DQEJ
ARYgc2VjdXJpdHlAYXRpeC5vcGVudGVsZW1hdGljcy5vcmcwHhcNMDgxMjI5MTk0
MDAxWhcNMDkxMjI5MTk0MDAxWjCBtzELMAkGA1UEBhMCQk8xDzANBgNVBAgTBkxh
IFBhejEPMA0GA1UEBxMGTGEgUGF6MR0wGwYDVQQKExRDZW50cm8gZGUgRGVzYXJy
b2xsbzEfMB0GA1UECxMWSmVmYXR1cmEgZGUgZGVzYXJyb2xsbzEfMB0GA1UEAxMW
ZGVzYXJyb2xsby5kb21pbmlvLmNvbTElMCMGCSqGSIb3DQEJARYWZGVzYXJyb2xs
b0Bkb21pbmlvLmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA2zUg9HJV
iflSTN0NDkaKTTIUEvcHUkjTSCqW4QUs8msWdsG1gfPsexUwAVhajdss09fi96TN
kfwb7vSfZJ+d5nLSNKefkBioMPOkftzYfWO5/Rl7cFqVnJPBvu2nFG6P4FasLkk2
39i/pw+Smb8W/XwLvKDTp8a+dX7YQlWCia8CAwEAAaN7MHkwCQYDVR0TBAIwADAs
BglghkgBhvhCAQ0EHxYdT3BlblNTTCBHZW5lcmF0ZWQgQ2VydGlmaWNhdGUwHQYD
VR0OBBYEFAQiHk+0uMsFRjFyPYqCRQZr0eRUMB8GA1UdIwQYMBaAFKrEOw3B17qD
MsSNLk8nXJv1wNxhMA0GCSqGSIb3DQEBBQUAA4GBAGEuZNU2LXpQrt4lze9gAvRS
LK6sZbDWqbOyMKQ2B62zLzLO6eqEXv0NtZd82HuZos7OZKk0AV6R831Qp35xkctd
Z2BGrCNNgazi/j0yVszDruB6pylXu6HLXmtxQyp46P6EE95yDy//eG3/aKn/htow
I2/x9Azkz+ZUAretnz4R
-----END CERTIFICATE-----
```

Esta misma información (de forma más entendible), también podemos obtenerla al probar el certificado mediante un browser, tal como se muestra en las siguientes imágenes:





Certificados comprometidos

Si un certificado en particular ya no se considera como confiable, por alguna sospecha de que haya sido comprometido, éste deberá ser revocado por la CA.

El proceso de revocación en sí es una tarea sumamente sencilla, la parte complicada es difundir la lista de revocación a los clientes y aplicaciones que hagan uso de este certificado.

Una CA bien configurada, almacena por seguridad una copia de los certificados, aspecto que facilita la comprobación de la huella digital (fingerprint) del certificado generado, una forma de comprobar es la siguiente:



Revocando un certificado

En el ejemplo siguiente procedemos a revocar el certificado correspondiente a **laboratorio.dominio.com**.

```
esteban@desarrollo:~/RevistaAtix
Archivo  Editar  Ver  Terminal  Solapas  Ayuda
[esteban@desarrollo RevistaAtix]$ openssl ca -revoke certs/laboratorio.dominio.com.crt -config open
ssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for /home/esteban/RevistaAtix/private/cakey.pem:
Revoking Certificate 02.
Data Base Updated
[esteban@desarrollo RevistaAtix]$
```

Lista de Certificados Revocados (CRL)

Como comentamos anteriormente la revocación de certificados es bastante sencilla, por lo que debemos otorgarle mayor esfuerzo al difundir la lista de certificados que fueron revocados, para esto debemos generar una lista de certificados revocados, como se muestra a continuación:

```
esteban@desarrollo:~/RevistaAtix
Archivo  Editar  Ver  Terminal  Solapas  Ayuda
[esteban@desarrollo RevistaAtix]$ openssl ca -gencrl -out crl/crl.pem -config openssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for /home/esteban/RevistaAtix/private/cakey.pem:
[esteban@desarrollo RevistaAtix]$
```

El comando anterior, creó el archivo **crl/crl.pem** cuyo contenido es el siguiente:


```
-----BEGIN X509 CRL-----
MIIBsTCCARoCAQEwDQYJKoZIhvcNAQEFBQAwgcExCzAJBgNVBAYTAkJPMQ4wDAYD
VQQIEwVPcnVybzEOMAwGA1UEBxMFT3J1cm8xFTATBgNVBAoTDFJldmlzdGEgQXRp
eDEoMCYGA1UECxMfRGlyZWNjaW9uIHkgQ29yZGluYWNpb24gR2VuZXJhbDEgMB4G
A1UEAxMXYXRpeC5vcGVudGVsZW1hdGljcy5vcmcxLzAtBgkqhkiG9w0BCQEWIHNl
Y3VyaXR5QGF0aXgub3BlbnRlbGVtYXRpY3Mub3JnFw0wODEyMjkyMDE1MDdaFw0w
OTAxMjgyMDE1MDdaMBQwEgIBAhcNMDgxMjI5MjAxMDI5WqAOMAwwCgYDVR0UBAMC
AQEwDQYJKoZIhvcNAQEFBQADgYEACob8f0zKaaKkDLkz7mLkxW20qulMmWxr/Xv2
GdeBgLvIt+NYyenkODO9wO+JgTu0ogkjZHHcvYvSoOtBoNnwHLjB1raCMT8PFB9/
B2aqPg5qRQrPCJTjddMYgj3WY6i8imWqO0BuoCStcZnmUDwLB7EAF+WNZH6lBTBJ
IoQpwhQ=
-----END X509 CRL-----
```


Esta información debe ser difundida y puesta en conocimiento a las entidades, personas y aplicaciones que usen nuestra CA; pero si deseamos podemos obtenerla en modo texto como muestra la figura siguiente:

```
esteban@desarrollo:~/RevistaAtix
Archivo  Editar  Ver  Terminal  Solapas  Ayuda
[esteban@desarrollo RevistaAtix]$ openssl crl -in crl/crl.pem -text -noout -CAfile cacert.pem
verify OK
```

La instrucción anterior muestra de forma más legible el contenido de la lista de revocación, como se ve a continuación

```
verify OK
Certificate Revocation List (CRL):
        Version 2 (0x1)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: /C=BO/ST=Oruro/L=Oruro/O=Revista Atix/OU=Direccion y Cordinacion
General/CN=atix.opentelematics.org/emailAddress=security@atix.opentelematics.org
        Last Update: Dec 29 20:15:07 2008 GMT
        Next Update: Jan 28 20:15:07 2009 GMT
        CRL extensions:
            X509v3 CRL Number:
                1
Revoked Certificates:
    Serial Number: 02
        Revocation Date: Dec 29 20:10:29 2008 GMT
    Signature Algorithm: sha1WithRSAEncryption
        0a:86:fc:7f:4c:ca:69:a2:a4:0c:b9:33:ee:62:e4:c5:6d:b4:
        aa:e9:4c:99:6c:6b:fd:7b:f6:19:d7:81:80:bb:c8:b7:e3:58:
        c9:e9:e4:38:33:bd:c0:ef:89:81:3b:b4:a2:09:23:64:71:dc:
        bd:8b:d2:a0:eb:41:a0:d9:f0:1c:b8:c1:d6:b6:82:31:3f:0f:
        14:1f:7f:07:66:aa:3e:0e:6a:45:0a:cf:08:94:e3:75:d3:18:
        82:3d:d6:63:a8:bc:8a:65:aa:3b:40:6e:a0:24:ad:71:99:e6:
        50:3c:0b:07:b1:00:17:e5:8d:64:7e:a5:05:30:49:22:84:29:
        c2:14
```

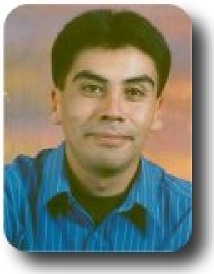
En las próxima entrega, mostraremos algunas formas útiles de manipular los certificados, algunos frontends, etc.

Referencias

[1] http://www.openssl.org/

[2] http://es.wikipedia.com

Autores



Esteban Saavedra López
Líder de la Comunidad ATIX (Oruro – Bolivia)
Activista de Software Libre en Bolivia
jesaavedra@opentelematics.org
http://jesaavedra.opentelematics.org



Joseph Sandoval Falomici
Profesor universitario
Entusiasta de Software Libre
josephsandoval@gmail.com

Willay news

Información Actual

Willay news

Google Chrome libero versión 1.0 y ya está la versión 2.0 en fase beta



Inesperadamente Google Chrome lanzó la versión 1.0 a mediados de Diciembre de 2008, Chrome que había anunciado su primera versión en Enero del 2009 al parecer se adelantó, esto debido a que probablemente tenga un plan estratégico de salir al mercado como un navegador oficial con los fabricantes OEM, me refiero a Sony, Toshiba, Hp entre otras.

Pero hablemos del nuevo navegador que según los empleados de Google, Chrome es uno de los navegadores que avanzan al ritmo de la evolución de la Web, es decir, nos ofrece mayor rapidez, estabilidad y Seguridad, además de que el código es abierto.

Según los usuarios que han descargado Chrome en la versión 1.0, sin duda han apreciado las mejoras y tras una serie de críticas, Chrome salió de la fase beta aunque falta mejorar el uso de extensiones como lo hace firefox y esta versión es sólo para Windows no descartando que ya se tendrá las nuevas versiones de Chrome para Mac y Linux a mediados del 2009.

No esperando más, en Enero sale la versión 2.0 mejorando las extensiones, el autocomplete, zoom, Javascript V8 actualizado y el soporte a script greasemonkey entre otras, por lo que se puede apreciar una velocidad notable en esta versión.

Mobile World Congress 2009 (MWC)

Uno de los eventos más importantes a nivel mundial es el Mobile World Congress que se llevará a cabo en Fira Barcelona del 16 al 19 de Febrero, donde muchos operadores y empresas importantes tales como Sony Ericcson, Hewlet Packard, Nokia, Microsoft y otros mostrarán al mundo su tecnología móvil y sus innovaciones.

Este evento reunirá a miles de ejecutivos que han confirmado su participación, 1300 expositores y 400 conferencistas entre ellos Steve Ballmer consejero delegado de Microsoft, César Alierta presidente ejecutivo de Telefónica , Chris DeWolfe cofundador de MySpace, Vittorio Colao Director ejecutivo de Vodafone, y representantes de las compañías AT&T, Nokia, NTT DOCOMO, Telenor y Verizon Communications.

Este congreso permitirá a muchos ejecutivos hacer negocios, compartir ideas, experiencias, tratar temas como la crisis económica, las tecnologías avanzadas de banda ancha móvil, Long-Term Evolution, el móvil abierto, etc.

Al parecer Apple será el ausente en ésta nueva versión de MWC, luego de haber tenido un importante éxito con el famoso Iphone 3G el pasado 2008, pero sin duda este evento traerá muchas novedades en tecnología, tanto de empresas conocidas y nuevas, probablemente encuentres el móvil de tus sueños en el MWC 2009.

Autor



Ivonne Karina Menacho Mollo
Titulada de la carrera de Ingeniería de Sistemas e Informática (F.N.I.)
Conductora programa radial “**Doble Clic**”
ivonnekarina2003@hotmail.com

Willay news

La campaña BadVista cierra con Gran Éxito

Hace tres años, Microsoft anunció una campaña de publicidad de 300 millones de dólares, siendo la mayor de todas en su historia, dirigidas a la promoción de Windows Vista, los activistas de Software libre sintieron venir el tiempo en el cual podría haber una masiva y súbita reducción de la libertad de los usuarios de computadoras.



Windows Vista se promocionaba con la etiqueta de no tener rival alguno a pesar de toda esa gran campaña de marketing, la llegada de Windows Vista fue sinónimo de "sistema fallido" sin embargo, Microsoft vendió muchas licencias de este S.O.

En diciembre de ese mismo año se inicia la Campaña BadVista con dos objetivos el de: "Exponer el daño infligido a los usuarios de computadoras por el nuevo Windows Vista y promover alternativas libres que respeten su seguridad y privacidad" dirigida por la FSF (Free Software Foundation).

Muchos se preguntaron en ese entonces ¿y cual es el problema con Windows Vista?

El experto Bruce Schneier explica: "Windows Vista incluye una serie de características que uno nunca quisiera, estas características harán que su computador sea menos confiable, menos seguro, menos estable y más lento. Causará problemas de soporte técnico, incluso se puede requerir que usted actualice algunos de sus periféricos de hardware y software existentes en su computadora, todo esto es porque existen DRM (Digital Rights Management que aplica prohibiciones al uso de los programas por cuestiones de derecho de autor) incorporados en Windows Vista"



Este año a principios de Enero, la FSF cierra la campaña con Gran Éxito teniendo como resultado a 7000 activistas registrados, 250 noticias describiendo los problemas que causa este sistema propietario y la DRM, contando con el gran apoyo de organizaciones no dedicadas al software como: Green Party, Friends of the Earth International, People & Planet, New Internationalist y Legambiente.

Esto muestra que empresas, universidades, gente dedicada al software o no, están aprendiendo a defender sus derechos de libertad. Como se dijo en la despedida de la campaña "…es evidente que Vista a perdido su oportunidad de ser adoptado universalmente…". El éxito de la campaña BadVista si fue un hito muy importante, de un camino muy largo a conquistar.

Nuevo smartphone: Palm Pre

Para los palmeros y gente en general, presentamos a Palm Pre ganador de dos importantes premios en la feria CES 2009 (Internacional Consumer Electronics Show).



El premio al mejor producto de la feria “Best in Show” y reconocimiento del público “People’s Voice Award” fueron otorgados al último logro de Palm un smartphone llamado Palm Pre, cuyas características serán la envidia de muchas compañías.

Con medidas de 5.9x10x16.9 cm. y un peso de 135 grs. cuenta con una pantalla multi-táctil de 3.1 pulgadas con resolución de 320x480 pixeles , un teclado físico deslizante QWERTY, con una plataforma hardware OMAP3 de Texas Instrument la cual será la encargada de procesar la información, según dicen una solución potente para un teléfono móvil.

Aquí esta lo mejor, su Sistema Operativo denominado WebOS esta basado en Linux, con una interfaz atractiva, te permite usar varias aplicaciones a la vez, su navegador Web te muestra las páginas a manera de cartas en vez de pestañas y se pueden aumentar de tamaño con los dedos… interesante verdad?, Palm Synergy será el encargado de sincronizar el sistema con Google, Facebook, Outlook , con soporte de mensajería instantánea y búsquedas integradas.

Y para los desarrolladores que están familiarizados con CSS, HTML, XML den por seguro que las aplicaciones de Palm Pre están hechas con estos lenguajes , no se tendrá que hacer un gran esfuerzo de adaptación, teniendo a la mano la posibilidad de programar aplicaciones para este smartphone.



Más características que pueden gustar mucho: Palm Pre cuenta con GPS, conectividad WiFi, Bluetooth, memoria de 8 GB, cámara de fotos de 3 megapixeles y flash LED.

Quedaron con la boca abierta? Palm Pre saldrá al mercado a mediados de este año Solo falta esperar cual será el precio de esta magnifica creación de los de Palm.

Autor



Marcia Estrella Velasquez Aguilar
Egresada de la carrera de Ingeniería de Sistemas (F.N.I.)
mevaguerreradelaluz@gmail.com

Willay news

Depurador de CO2

Miles de personas al rededor del mundo nos vemos afectados por la contaminación del medio ambiente, principalmente en las grandes ciudades, donde existen una gran cantidad de automóviles, industrias, etc; aspecto que hace que la presencia del dióxido de carbono sea cada vez más elevado.

eCO2 (Dióxido de Carbono Scrubber) es un dispositivo que es investigado por el equipo dirigido por David Keith. Este dispositivo realiza la captura de CO2 directamente desde el aire y hace la limpieza del mismo, expulsándolo una vez depurado.



Puede ser que la tecnología no sea una novedad, y es más algunos escépticos no creen en la misma, pero el eCO2 es un depurador que intenta borrar la huella de carbono personal de la persona.

Es un concepto que aborda que una persona contribuya con su granito de arena en bien el medio ambiente, es un dispositivo que intenta convertirnos en ciudadanos más responsables en nuestro planeta.

Sus características principales son:

- ✔ forma de reloj deportivo de pulsera
- ✔ utiliza energía cinética como fuente de energía limpia



Esta demás decir que es un dispositivo al margen de ser innovador, es un dispositivo que intenta de alguna forma contribuir al cuidado del media ambiente.

Módems USB de alta velocidad

Actualmente la mayoría de los dispositivos tiene una forma de interactuar mediante el puerto USB (1.0, 2.0, 3.0), dispositivos como: wi-fi, bluetooth, Irda; pero la tecnología cada día va creciendo y ahora nos da la posibilidad de contar con módems USB de alta velocidad, como es el caso del módem EVDO 598U



El EVDO Rev-A. proporciona a los usuarios de dispositivos compatibles con velocidades de descarga promedio entre 600 kbps y 1,4 Mbps y velocidades de carga promedio de 350 a 500 kbps.

Este dispositivo también incorpora un receptor GPS integrado, así como una ranura de expansión microSD que admite tarjetas de memoria de hasta 32GB, y que efectivamente mejora la funcionalidad del módem; que a la vez funciona como una unidad flash USB.

Otro dispositivo de estas características es el módem U300 EVDO Rev-A AND WiMAX



Este dispositivo ofrece banda ancha inalámbrica 3G y 4G tanto, es un dispositivo dentro la revolución de banda ancha móvil, dando a los clientes lo mejor de ambos mundos.

Autor



Jenny Saavedra López
Diseño y Edición Revista Atix
jennysaavedra@gmail.com


Software Libre
Disciplina - un elemento importante
Revista de Software Libre ATIX
2008
ATIX 2008
WALL·E
Los robots
en el futuro
seguimos
usando
Software Libre

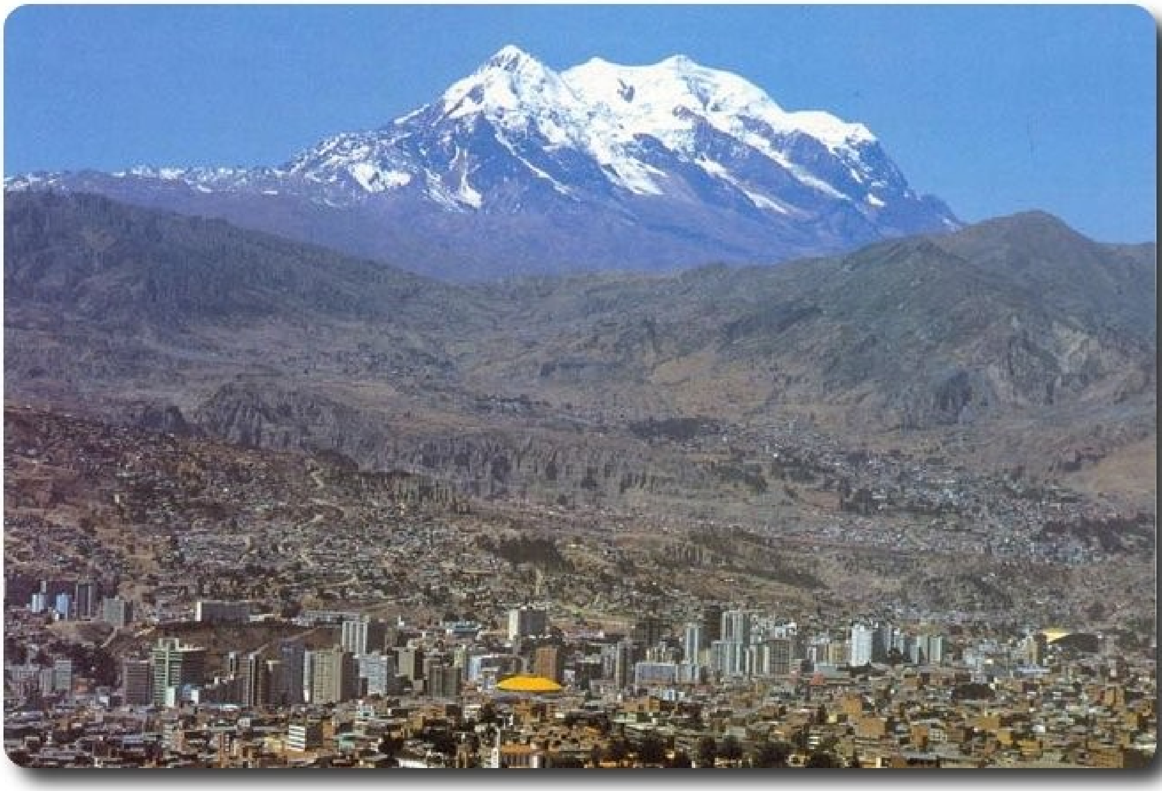

Position 1 - Patching the Kernel
Position 2 - Backend Arch
Position 3 - Downloading the Drivers
Position 4 - Mounting the Hard Disk
Position 5 - Congress of the GNU
Position 6 - Spanning-Tree Protocol Posture
Position 7 - Embedding the Stack
Position 8 - 01000101 = 45 = 69
Position 9 - Forking the Code
Position 10 - Loading the Queue
Position 11 - Partitioning the Drive
Position 12 - Piercing the Firewall



Conociendo lo Nuestro

La Paz

Turismo y Libertad



Vista de la ciudad con el fondo del majestuoso Illimani



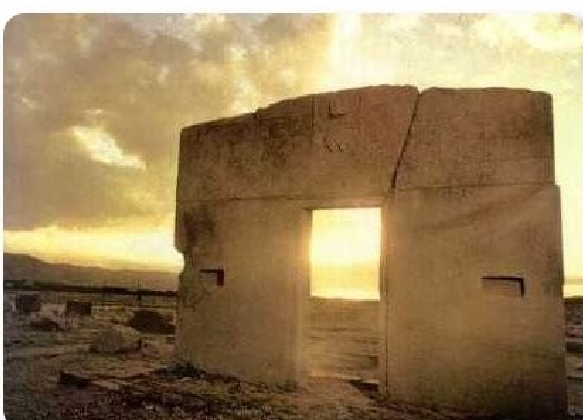
Vista del centro de la ciudad



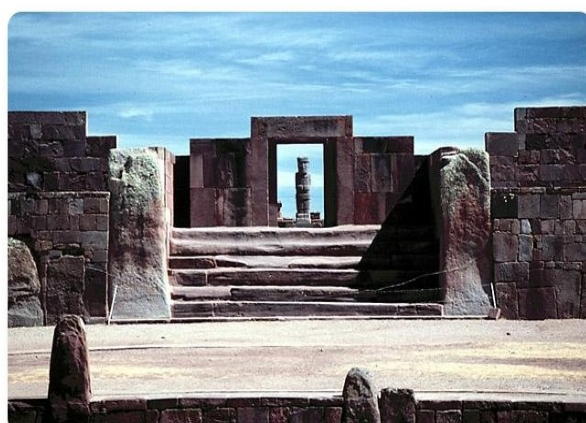
Paseo turístico por la ciudad de La Paz



Balsa de totora en el lago mas alto del mundo “El Titicaca”



Puerta del sol - Tiwanaku



Templo de Calasasaya - Tiwanaku

Libres para pensar, libres para decidir, libres para crear

Arte Libre

Te ofrecemos este espacio para mostrar tu Creatividad



Envíanos tus diseños y creaciones para publicarlos


ATIX 2008
Software Libre
WALL·E
El Software Libre perdurará en el tiempo

Consultas

Contacto

Para solicitar cualquier información, puedes contactar a:

- ✔ Esteban Saavedra López (jesaavedra@opentelematics.org)
- ✔ Jenny Saavedra López (jennysaavedra@gmail.com)

Publicación

Te invitamos a ser parte de la **Revista ATIX**. La forma de participar puede ser enviándonos:

- ✔ Artículos referidos a áreas como:
  - ✔ Instalación y personalización de Aplicaciones
  - ✔ Scripting
  - ✔ Diseño gráfico
  - ✔ Programación y desarrollo de aplicaciones
  - ✔ Administración de servidores
  - ✔ Seguridad
  - ✔ y cualquier tema enmarcado dentro del uso de Software Libre
- ✔ Trucos y recetas.
- ✔ Noticias.
- ✔ Comics.
- ✔ Links de interés.

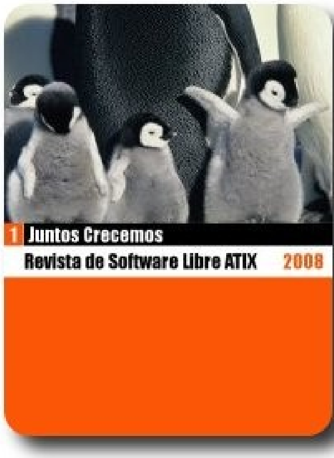


Bitacora

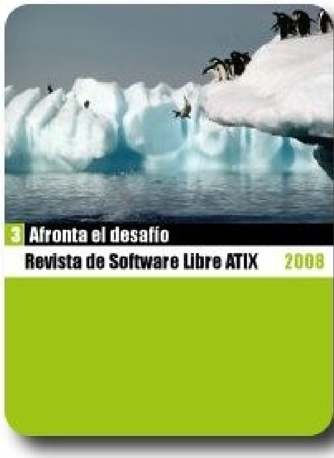
Empezamos a registrar nuestra historia:



28 de abril del 2008



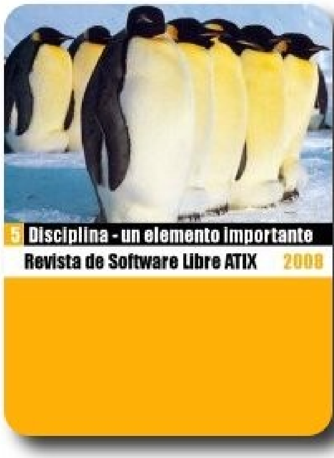
20 de junio del 2008



6 de agosto del 2008



17 de septiembre del 2008



10 de noviembre del 2008



17 de diciembre del 2008

ATIX

Marcamos Huella

http://atix.opentelematics.org